QATAR UNIVERSITY

COLLEGE OF ENGINEERING

PRIVACY-PRESERVING DECENTRALIZED INTRUSION DETECTION SYSTEM FOR IOT DEVICES USING DEEP LEARNING

BY

ALIYA TABASSUM

A Dissertation Submitted to

the College of Engineering

in Partial Fulfillment of the Requirements for the Degree of

PhD in Computer Science

January 2022

COMMITTEE PAGE

The members of the Committee approve the Dissertation of

Aliya Tabassum defended on 22/11/2021.

Dr. Mohsen Guizani
Dissertation Supervisor

Dr. Aiman Erbad
Dissertation Co-Supervisor

Dr. Abdelaziz Bouras
Committee Member

Dr. Wadha Lebda
Committee Member

Dr. Cagatay Catal
Committee Member

Dr. Mohamed Mahmoud
Committee Member

Approved:

Dr. Khalid Kamal Naji, Dean, College of Engineering

ABSTRACT

Tabassum, Aliya, PhD: January: 2022, Doctorate of Philosophy in Computer Science

Title: Privacy-preserving Decentralized Intrusion Detection System for IoT Devices using Deep Learning.

Supervisor of Thesis: Mohsen Guizani, Aiman Erbad.

The convergence of advanced networking, breakthrough distributed systems technologies, and smart services has rapidly expanded the threat landscape for IoT devices. Researchers have been looking into lightweight and adaptive technologies to solve the problems of cybersecurity in dynamic smart IoT systems, as these domains are increasingly targeted by cyber-criminals. In most of the scenarios, a peripheral defense, Intrusion Detection System (IDS) is proved effective to protect IoT devices. However, existing intrusion detection techniques have centralized designs with repetitive pre-processing steps, privacy leaks due to raw data exchange, and computationally expensive workloads for the resource constrained IoT devices. In this dissertation, we propose using Deep Learning (DL) and relevant distributed Artificial Intelligence (AI) techniques to develop an efficient and secure distributed IDS model. First, we demonstrate that effective pre-processing of input data greatly reduces the burden on the classifier and enhances accuracy in incremental distributed learning. The first contribution in this dissertation proposes a novel pre-processing technique, which ensures privacy of data of the IoT devices, eliminates redundancies, and selects unique features by following innovative extraction techniques. Our privacy-preserving incremental AI-based IDS can tackle zero-day attacks, particularly mutations of existing attacks. Second, the data imbalance

issues in intrusion detection can degrade the model accuracy, particularly in rare classes. To this end, Generative Adversarial Network (GAN) is effective in data augmentation to balance the available training data. The second contribution in this dissertation models the proposed distributed IDS in an innovative manner using Federated Learning (FL), which minimizes the data sharing to enhance privacy and performance. Our approach "FEDGAN-IDS" uses FL and GAN to effectively detect cyber threats in smart IoT systems. This is achieved by distributing the GAN network over IoT devices for training the model using local data and handling the model's distribution using FL. Overall, this dissertation proposes a privacy-preserving distributed IDS for IoT devices suitable for real-time protection scenarios. We evaluate our work using accuracy, delay, and other critical criteria using multiple datasets, such as NSL-KDD and KDD99. The model performs better and converges faster than the state-of-the-art standalone IDS models.

DEDICATION

To my husband for his support and trust in me. To my beloved children Abdul Rahman and Amatunnoor, and my mother for their love.

ACKNOWLEDGMENTS

All praise to Allah the Almighty who grant me all the blessing, the strength and endurance to complete this study.

I would like to sincerely thank my Thesis supervisors, Dr. Mohsen Guizani and Dr. Aiman Erbad, for encouraging me to deliver my best and assisting me at every single step. I am grateful for their positive approach and excellent guidance throughout my thesis. I owe a debt of gratitude, as this journey would not have been possible without their inspiration and critiques. I would also like to thank Dr. Amr Mohamed for his invaluable insight and feedback at critical times. My great honour is also bestowed upon all those faculty members who taught me various courses during my PhD and bolstered my knowledge.

Not least of all, I would like to express my unrestrained appreciation to husband, Mohammed Zubair and my family for their indispensable support during this entire phase. I extend my acknowledgement to my friends and colleagues who held me up on several occasions.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER 1: INTRODUCTION

Internet of Things (IoT) consist of Internet Protocol (IP) connected heterogeneous tiny objects in a hybrid network, which sense and communicate with internal or external entities to perform a certain service. It is a system of interconnected computing devices, humans and objects with unique identifiers capable of transferring information without human-human interaction and human-computer interaction. IoT technologies include Radio-Frequency Identification (RFID), Machine-to-Machine (M2M) communication, Wireless Sensor Networks (WSN), and Low power Wireless Personal Area Networks (LoWPAN). The Internet of Things has changed the perception of life in every sector of our life. The number of IoT devices is increasing everyday, and transforming the traditional objects into smart, and intelligent that can coordinate for decision making. The number of IoT devices surpassed the human population in 2010, and it is recorded 7 billion in 2018 [1]. Moreover, researchers predict that in 2025 this number may reach 22 billion with the expected economy generated by various IoT application domains to have a value between 4 and 11 trillion dollars [2]. Figure 1.1 shows the various IoT application areas, which include: smart power-grid, smart retail, smart supply chain, smart agriculture, smart industry, smart transportation, smart health, smart wearables, smart housing & buildings, and smart city.

Research demonstrates the power of IoT systems, anticipating a world with automatic detection and control of smart environments, such as the consequent unlocking of a door on the approach by an identified member while staying locked for unidentified individuals [3]. Nowadays, IoT devices are embedded with AI and Machine Learning (ML) techniques to ease the tasks of humans, making IoT devices, as intelligent agents, capable of decision making and performing actions on their own. IoT devices have

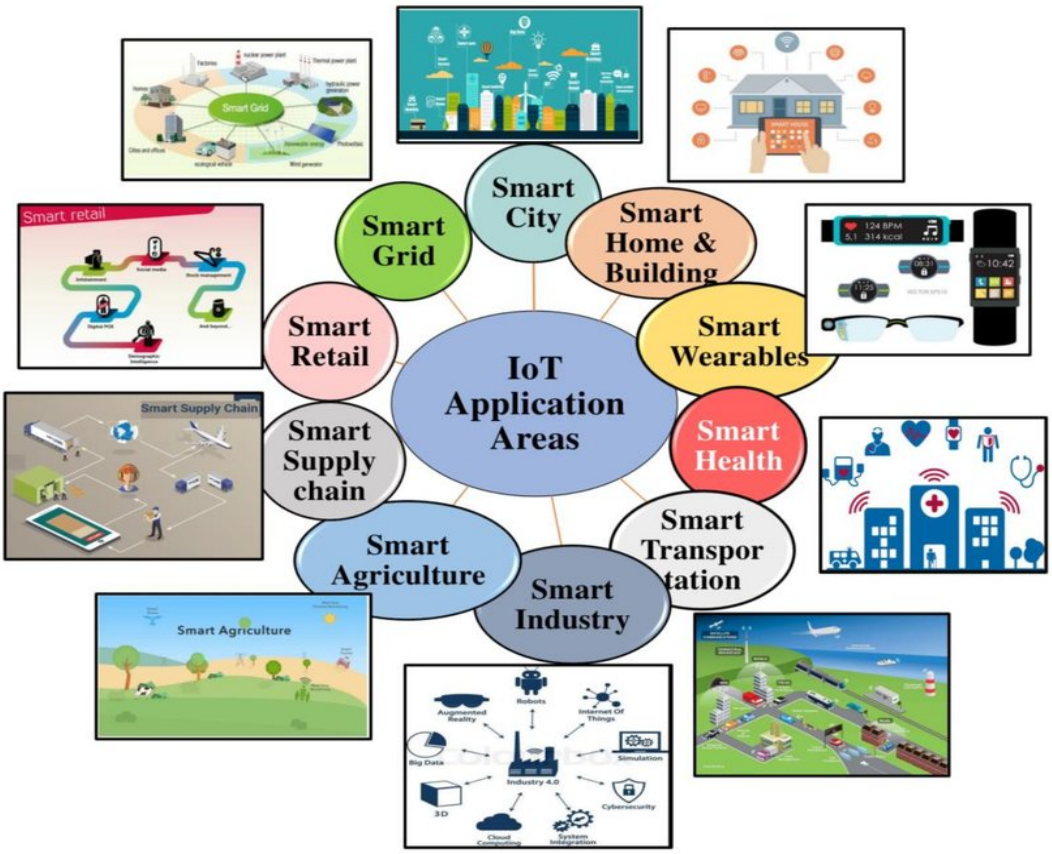

Figure 1.1: IoT Application Areas.

become so sensible and discerning that they are capable of envisioning the life-cycle, capability and efficiency of any product in any field. These devices are conspicuous, locatable, addressable and controllable through the Internet. IoT devices are capable of monitoring heart impulse and reporting abnormal heart rate and can be used to monitor the level of insulin in the body [4]. IoT devices are also being adopted in automobile industries with built-in sensors to alert the drivers when the tire pressure is low to avoid accidents [5].

Though IoT devices are rapidly being adopted worldwide, there are many privacy and security issues in different layers of the IoT architecture [6]. A report published by Hewlett-Packard Development Company (HP), as a part of the Open Web Application

Security Project (OWASP), has shown that security aspects are not taken into consideration while manufacturing these devices [7] [8]. These security threats are concerning and create obstacles to the services that IoT systems provide. The significant security challenges in IoT devices include privacy, data storage, secure communication, authentication and access control [9]. Applying appropriate defense mechanism, e.g., mitigation is necessary to thwart the cyberattacks before causing a great impact on the devices and end-users. Although the ever-increasing attacks cannot be mitigated fully, real-time network activity monitoring using an IDS, strong access control, and authentication can be adopted to prevent and detect attacks.

Motivation

The security of IoT devices is critical since the IoT networks are different from Cyber Physical Systems (CPS) and Wireless Sensor Networks (WSNs). These intelligent smart devices bring the risks from industrial control system space to different application domains affecting real life, leading to significant attention from researchers and security specialists all over the world. A shift to digital transactions in different applications mandates a shift in our mindset to secure all these transactions as they traverse small IoT sensors. The longer-range wireless communications, and resource constraints of battery, processor and memory makes the systems more complex. The privacy and security of IoT applications are concerned with confidentiality, integrity, and availability of the information and/or services. Meeting these security goals is significant for modern organizations to ensure trust, and to guarantee the safety of data [10]. Finance, usability, resources and many other factors influence the level of security mechanisms that are implemented in an IoT device. Every sector has to implement the security best practices

appropriate to their needs. The traditional mechanisms for cyber attacks detection, especially for known attacks are effective in certain situations but cannot be applied to its variants or absolute unknown attacks. The devastating attacks against IoT devices mandate a peripheral defense, IDS to ensure reliable and secure networking in IoT devices.

Security challenges, like undeveloped network standards, resource constraints such as power capacity, limited storage and low computational capability, prevent the deployment of various security solutions in IoT devices. Besides, with the increase in the number of devices, newer attacks are emerging. In literature, there are many IDS models have been proposed using AI techniques. However, the existing ML, especially DL models are computationally heavy and based on a strong assumption that the traffic samples available are sufficient, labelled and useful for model training [11]. IDS demands robust and improved techniques to survive against cutting-edge malicious attacks. Nevertheless, the zero-day attacks, groundbreaking attack techniques and eccentric hackers make any IDS outdated with the novel attacks. To defend against such disruptive malicious activities, the IDS needs continuous improvement at the pace of the variability in the traffic patterns. Therefore, we aim to propose a real-time IDS for IoT devices that is suitable for its resource restrictions and efficient in identifying attacks accurately.

Thesis Objectives and Contributions

The goal of this research is to develop an IDS for IoT systems to detect popular security threats with lower False Positive (FP) and True Negative (TN) rates. We aim to develop a real-time and effective IDS that learns the normal network activity from the regular network usage and creates a model using DL. The network traffic is monitored

intelligently based on the developed model and is categorized as malicious or benign. The research questions that are addressed in this dissertation are as follows: (1) What are the challenges and issues of existing IDS models that need to be solved? (2) How pre-processing tasks affects the DL model training? Is it possible to capture newer features from the incoming IoT traffic using efficient pre-processing techniques? To address this, a favourable pre-processing technique for DL model training is proposed such that it captures newer features from the incoming traffic. (3) What are the affects of a centralized IDS on the time and computational complexity of the model? After analysis of these critical metrics, we distribute the intrusion detection process on different neural networks. (4) How can we identify the rare classes samples efficiently? Which of the existing AI techniques can be used to preserve the privacy of data in decentralized way? For this, we design a privacy-preserving and scalable intrusion detection method using FL and GAN to overcome centralization and data imbalance issues, respectively.

We begin this dissertation by studying various intrusion detection models to identify the appropriate approach for protecting IoT networks. Existing anomaly-based and DL based approaches are examined to address the limitations and issues of the latest IDS models. Then, we build an IDS using heuristic pre-processing techniques and incremental learning. The proposed distributed IDS reduces the overhead on the centralized Edge classifier such that the subsequent latency between pre-processing and decision making phases is minimum. The main contributions of this work are summarized as follows:

1. **We propose a efficient pre-processing technique** for IoT devices using DL, such that the detection process has less computational complexity without consuming much resources or degrading the network performance. Our proposed parallel pre-

processing technique utilizes the computational power and memory resources of the IoT devices to ensure the security and privacy of the raw data. Sharing the pre-processing task among IoT devices reduces computational resources and delays at the source, and gives comparable performance to the centralized classification task.

2. **An incremental learning model** is proposed to update the classifier seamlessly with emerging features in order to detect new attacks. Our incremental model gives better accuracy than other approaches.

3. **A comprehensive evaluation** of the incremental intrusion detection model using different experiments with standard datasets and real-time IoT traffic for Denial of Service (DoS) attack detection is provided. We also perform decision time analysis for the classification process and compare it to other centralized models.

In the second phase, we model the proposed IDS using FL and GAN to tackle data imbalance and privacy issues. The contributions to this end are as follows:

1. **We designed a novel distributed GAN-based intrusion detection model** for smart IoT systems. The GAN generated synthetic data augments the data on an IoT device to train the intrusion detection model individually. The GAN network solves the problem of limited, missing and imbalanced data on the IoT devices.

2. **We have proposed a privacy-preserving FL framework** allowing multiple smart IoT devices contribute to building a global intrusion detection model. Each device trains its single model on its own data and synthetic local data generated by the local GAN and transfers the model parameters to the global model. The local and generated data of each device are not shared with other IoT devices. This task

ensures data privacy by performing the data pre-processing and model building at each device. The global model performs parameters aggregation and distributes the updated model to IoT devices.

3. **We developed a binary and multiclass classification** intrusion detection classifier after multiple FL rounds of communication with available IoT devices in the network.

4. **Finally, we evaluated the performance of the distributed intrusion detection model** in terms of accuracy and other metrics with two standard datasets; NSL-KDD and KDD99.

Thesis Overview

The structure of the rest of the dissertation is as follows: Chapter 2 provides background and related works, in which we discuss the types of IDS and existing intrusion detection models in the literature. In Chapter 3, we elaborate on existing IDS and its challenges, specifically for IoT devices. Chapter 4 presents a heuristic pre-processing technique using incremental learning that reduces computational overhead for intrusion detection model. In Chapter 5, we explain our privacy-preserving intrusion detection model using FL and GAN. Finally, the dissertation is concluded with future research directions and conclusions in Chapters 6 and 7, respectively.

CHAPTER 2: BACKGROUND AND RELATED WORK

Intrusion Detection System

IDS monitor and protect a network from malicious activities or policy violation. Intrusion detection is the process of detecting unauthorized access and intrusions in the network and information systems [12]. Intruders can be internal or external. Internal intruders are the legitimate users who try to escalate privileges in order to access unauthorized data or services, whereas, external intruders are the people outside the network who attempt to gain access to the network and/or information systems. There are two types of IDS: network-based intrusion detection system (NIDS) and host-based intrusion detection system (HIDS).

Network-based IDS is connected to one or multiple network segments and monitors inbound and outbound traffic for malicious activities. Traditional NIDS mechanism is challenging and restrictive to be applied to IoT devices due to heterogeneous connectivity, constrained resources, and limited power. The nodes in traditional systems monitor inbound and outbound traffic without any resource or bandwidth constraints. In IoT devices, many NIDS mechanisms have been developed using attack signatures but the False Positive rates are higher, i.e., false alarms and unknown attacks cannot be identified [13]. Host-based IDS is connected to any component like computer device, node, or a router to monitor network traffic on that particular device. Unlike NIDS, HIDS scans operating system processes, file system modifications, and system calls [14]. HIDS is deployed on a specific device and protects that device from internal and external attacks. Nowadays, IDS is designed to protect devices and also networks. The hybrid way of combing HIDS and NIDS helps to protect the entire IoT system, which is chosen as per

the need of the system.

The principal challenge in IDS is the identification of anomalous patterns in the network. Sometimes, a benign behavior is detected as malicious and vice versa. The number of FP and TN should be minimum to reduce the error rate in anomaly detection [15]. This challenge emerges due to the complexity of the networks that are connected and numerous devices that are interacting and exchanging information in IoT systems. These abundant distributed devices that are connected through the Internet also give different ways to launch Distributed Denial of Service (DDoS) attacks, making them unusable for a point in time or crashing the devices. Unlike user-driven computer networks, IoT networks are object-driven, so it is difficult to apply conventional computer networks IDS mechanisms to IoT networks. Therefore, specialized mechanisms are needed to monitor, secure, and manage IoT devices and networks from evolving threats and vulnerabilities.

IDS Placement Strategies

The placement of the IDS is significant to ensure higher rates of detection. Here, we have provided different ways of placing an IDS in an IoT network.

1. *Centralized:* In this case, the IDS is placed at any centralized component either at the border of the node or at any host. When the IDS is placed at the border router, it is able to analyze all the traffic between the node and the Internet, while the traffic across the border router is left unmonitored. Moreover, when a part of the network is compromised, the centralized IDS may not monitor those nodes during the attack.

2. *Distributed:* In this mechanism, an optimized IDS is placed at each physical object

meeting the resource constraints of the nodes in IoTs. A distributed lightweight IDS is proposed using the packet payloads and attack signatures in which the IDS is placed in various places to cover the whole network monitoring [16]. To reduce the number of matches, auxiliary shifting and early decisions are adopted. Lee et al. have proposed a anomaly-based model with distributed IDS, which proved to be successful to some extent in detecting intrusions [17].

3. *Hybrid Approaches:* This scenario integrates the different IDS approaches with the placement strategy to maximize the advantages and minimize the limitations. Hybrid IDS placement bolsters the performance when centralized and distributed schemes are combined together.

From all the above methods, a hybrid approach, which suits the situation and network structure can be adopted. It is impractical to design a hybrid approach specifically for each different scenario. Hence, researchers have focused on ML and AI techniques, specifically on DL algorithms to design accurate IDS for IoT networks.

Intrusion Detection System Approaches

IDS approaches are categorized into signature-based, anomaly-based, specification-based, and hybrid based on the detection approach. This section discusses the different types of IDS approaches.

Signature-based Detection

This approach recognizes anomalous patterns based on the signatures of the attacks stored in the internal database of IoT devices or IDS. Whenever an attack signature

matches with the database, an alert is triggered. This process is very effective and fast for identifying known attacks. However, it is difficult to detect new attacks or the attacks for which signatures are not stored or even variants of known attacks[18]. Deploying any IDS in IoT devices on the low capacity nodes and low power networks is highly difficult and challenging. In signature-based methods, both the cost of storing the signatures in the databases and the computational cost of running learning algorithms for checking each signature are high. A signature-based IDS, Intrusion Prevention System (IPS), and Network Security Monitoring (NSM) method using Suricata Engine are adapted for (IPv6 over Low-Power Wireless Personal Area Networks (6LoWPAN) based networks [19]. An improvement to this mechanism is developed to reduce the computational cost of comparison between attack signatures and network packet payloads [16]. However, the signature-based approach is unsuitable for zero-day attack scenarios.

Specification-based Detection

This method sets guidelines for the expected behavior of the network components like nodes, routing tables, and protocols. The purpose is similar to anomaly detection, i.e., when the behavior is deviated from the specification, it is considered an intrusion. However, unlike anomaly detection, this approach needs a security expert to define the specifications for the elements and this procedure guarantees lower False Positive rates. No learning algorithms are needed, but the challenge is that the different specifications required for different platforms or environments [20]. One such approach is implemented to tackle denial of service attack in which the maximum capacity of each middleware layer is defined beforehand and if the number of requests matches or exceeds the capacity, an alert is triggered to the network administrator [21]. One of the specification-based

approaches is proposed for Routing Protocol for Low-Power and Lossy Networks (RPL) where the behavior of the protocol is fed into a finite state machine to monitor the network intrusions and malicious behavior [22]. As an extension work by Le et al. [23], simulation trace files are used to generate finite state machines. Most of the manually deciding specification approaches are highly dependent on the expertise of the security team and network administrator. Inappropriate specifications result in higher false alarms and in turn, increase the risk of network security.

Anomaly-based Detection

Anomaly-based model detects unknown attacks and often relies on ML algorithms to create a model of trustworthy 'good' traffic activity, and then, compares network activity against this model. Sometimes, the previously unknown complex legitimate activity is classified as malicious, i.e., False Positive. The ML techniques and statistical methods used for matching algorithms are heavy to be suitable for deploying on low capacity nodes, which is one of the challenges that needs to be considered. The authors in [24] have proposed an anomaly-based method for detecting botnets based on the average of three metrics: TCP control fields sum, number of connections for each sensor, and packet length. They analyzed the nodes behavior for identifying anomalous activity and considered energy consumption of the node as a parameter. They established models of conventional energy consumed by the nodes in normal routing and if any node is abnormal in power consumption, then it is removed from the 6LoWPAN routing table [17]. Another anomaly detection mechanism for resource constrained IoT devices is proposed by Summerville et al. [25]. The authors claim that the protocols in IoT devices are simple leading to similar network payloads. They perform feature selection

using bit-pattern matching. Likewise, one of the efficient methods is devised in 2015 by Pongle et al. for detecting wormhole attacks in IoT networks, which consumes very low power and energy [26]. The approach is based on the number of packets shared between nodes. If the packet exchange rate is high compared to the normal behavior, then an alert is triggered. However, defining a model based on few features is not efficient as it detects only defined attacks and results in higher FP rates for other behaviors, creating obstacles for normal network activity. Anomaly-based detection with appropriate feature extraction and lightly loaded learning techniques are suitable for current IoT networks. Anomaly-based detection integrated with DL techniques can be classified into supervised, unsupervised, and semi-supervised learning-based detection.

Intrusion detection approaches based on Artificial Intelligence

Artificial Intelligence has been used extensively to solve security problems as it has the capability to learn, deduce, and decide based on cognitive functions of pattern recognition. In this section, we present IDS approaches using various AI techniques in literature.

IDS using Machine Learning

The effectiveness of ML techniques in fraud detection, image recognition, and text classification has encouraged security researchers to employ these algorithms for anomalous pattern detection and identify abnormal behaviors to enhance the security of IoT networks [27]. The ML algorithms rely on patterns from data sets taken as inputs. For this reason, ML is applied even in conventional methods of attack detection such as

signature-based and anomaly-based in traditional Internet networks [28]. ML algorithms have been used for data processing and management in IoT devices to extract useful data from the voluminous data [29]. An ML-based distributed attack detection method for IoT devices in Fog networks is implemented using Extreme Learning Machine (ELM) classifier integrated with Semi-Supervised Fuzzy C-Means for efficient detection in lesser time [30]. This approach is tested using popular NSL-KDD dataset. Similarly, Shiven et al. [31] have proposed an integrated IDS using ML and anomaly-based detection approach. This method provides security as a service incorporating different communication protocols and monitors inbound-outbound traffic. It easily adapts to different network topologies and does not require any special hardware for the setup in the network.

Among the methods of developing IDS using ML that are discussed in literature, only few have been evaluated in real-time, specifically for IoT. Most of these intrusion detection models using intelligent techniques are customized for Wireless Sensor Networks or for the traditional Internet architecture. Moreover, ML techniques are ineffective in detecting attack variants and nowadays, some attack types are being modified several times to hide its actual signatures or behavior. Therefore, DL is recommended for IDS for hidden parameters extraction capability, which enables it to easily identify multiple variants of attacks. Also, these methods collect data at centralized unit to make informed decisions [32], which is not a recommendable solution as it threatens privacy. Nevertheless, new directions of AI using DL have a decentralized and distributed way of processing, analytics, and decision making on big data [33].

IDS using Deep Learning

IoT systems support a diverse protocol stack, which causes numerous zero-day attacks to emerge. It is harder for a traditional ML mechanism to detect the small mutations of attacks over time. DL is considered as a major step forward in Artificial Intelligence. It is inspired by the ability of the human brain to adapt to circumstances and deduce from past experiences. DL methods are known for their success of high-level feature extraction capability in big data due to its deep structure of layers. The hidden layers of Deep Neural Network (DNN) are shown in Figure 2.1. DNN layers can identify hidden patterns from the training data and completely rely on recognizing the true face of any variant. Deep networks are far better than the shallow ML methods in attack detection with high accuracy [34]. DL is a resilient approach for current IoT networks to detect small variants of the attacks. Besides, the compression capabilities and unsupervised pretraining are the key features of DL, making it possible to be deployed on IoT constrained networks IDS.

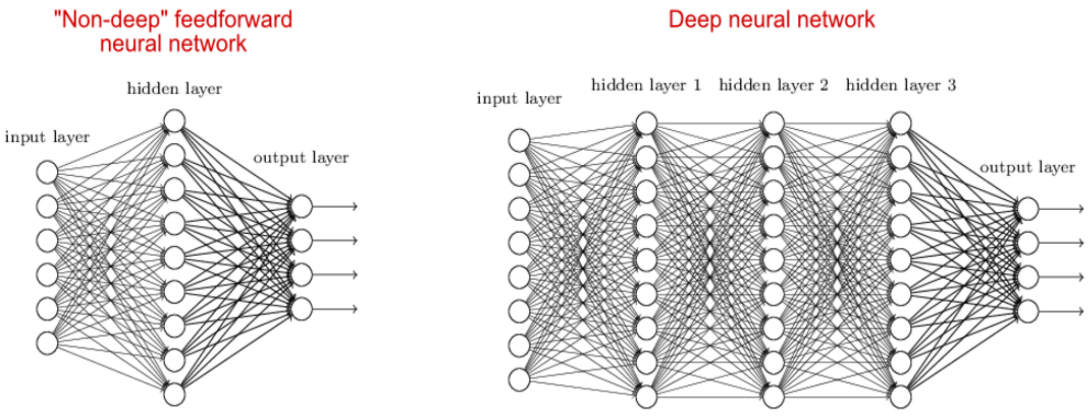

Figure 2.1: Non-deep and Deep Neural Networks.

Sakurada et al. [35] have adopted DL network, called AutoEncoder (AE), for

anomaly detection, where the normal network behavior is self-learned by the auto-encoders. The approach recorded abnormal (different) results for the same dataset during training. This model can be enhanced by choosing appropriate hyper parameters for the network. Another distributed intrusion detection approach using DL is proposed for Social Internet of Things (SIoT) [36]. The authors have shown that the distributed IDS in Fog network is more scalable than the centralized cloud. This method can be evaluated by using various datasets, payload data, and algorithms for comparisons and future enhancements using existing models.

Generative Adversarial Network

GAN was initially introduced by Ian GoodFellow [37] for estimating generative models. The GAN network is computationally heavy, as the architecture consists of two deep neural networks called the generator (G) and the discriminator (D). In the traditional GAN network, G and D are tightly coupled to reach a target learning rate. The generator is trained to produce artificial data while the discriminator is trained to differentiate the original and generated data. The two networks run in parallel to improve their performance gradually. After n iterations, the generator learns to output data close to the original data and the discriminator learns to identify the source of the data, i.e., real input or generator sample. This is like a min-max player game framework, which can be represented by the function below.

$$\min_{G} \max_{D} = [E_{x \sim P_{data}}[logD_i(x)] + E_{z \sim P_z}[log(1 - D_i(G_i(z)))]] \tag{2.1}$$

$G_i(z)$ is the synthetic data that the generator produces and the discriminator output is $D_i[0, 1]$, which shows the probability that the data is real or duplicate. The objective

of the generator is to minimize the probability of identifying the source of data by the discriminator, i.e., $(1 - D_i(G_i(z))$ while the discriminator try to maximize the probability of source identification, i.e., $(1 - D_i(G_i(z))$ and $D_i(x)$. These networks are powerful in data augmentation, data simulation, anomaly detection, image generation, and text-to-image translation [38]. The architecture of GAN is shown in Figure 2.2.

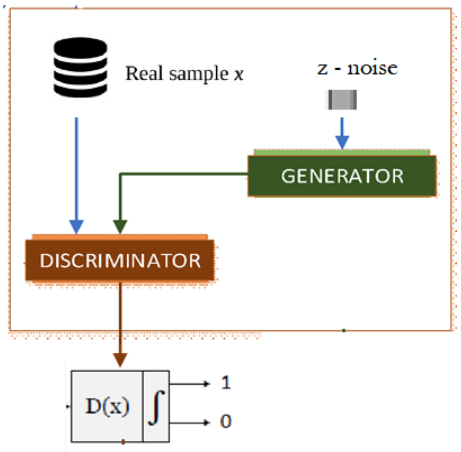

Figure 2.2: GAN Network.

Mostly, the GAN has been used in its typical network composed of two neural networks. The input of the generator is a noise signal, which is a random vector of size *k* that follows a normal distribution and outputs data similar to the training data. The input to the discriminator is fed from either the generator or training sample to differentiate it. Some of the recent researches [39] [40] have modified the GAN architecture into a distributed framework to improve its convergence rate.

CHAPTER 3: CHALLENGES OF INTRUSION DETECTION SYSTEM FOR IOT DEVICES

In this chapter, we review the challenges to develop an IDS for IoT devices. We also enlist the limitations of the existing IDS models in the literature.

Intrusion Detection in Streaming IoT Data

Classification of streaming data is difficult due to the dynamic nature of the incoming traffic. This is a major problem in ML and DL applications. The learning algorithms are trained at one time but need to be retrained due to the continuously evolving nature of data. retraining requires huge time overhead, computational resource requirements, and memory footprint due to the large scale IoT datasets arriving in real-time from the environment. While many learning algorithms can work with the raw input features, their performance degrades as the number of samples grow. The performance degradation and the inability to classify is due to the high number of features that have to be processed for decision making, which is referred to the "curse of dimensionality" [41].

Feature selection is one of the approaches to reduce input features, i.e., values given to the classifier. Usually, feature selection techniques analyze each variable independently. Sometimes, the variables that do not provide any information individually, give useful information when they are combined with the other variables. Therefore, feature extraction methods are proposed to construct meaningful features or to extract high-level information from the raw features [42]. One such way is data compression in which the number of bits that represent the data are reduced. AEs are unsupervised neural networks that perform data compression. AE is a simple neural network that reconstructs input into output with the least possible distortion, i.e., the output is close to the input.

It applies backward propagation, similar to Artificial Neural Network (ANN), to set the target values to be equal to the inputs. AEs can be used for supervised or unsupervised learning as it can handle data that is unlabelled but labeled data has more information that can be used.

AEs are efficient in reducing noise, dimensionality reduction, and learning important features in the data while reconstructing the input, i.e., pretraining. The learned features reveal the non-linear properties of the data. Extracted features provide a good discriminative ability for the classification task [43]. The input features are crucial parameters for efficient intrusion detection. Different features change the detection performance of the intrusion detection model. There are standard algorithms for dimensionality reduction, such as Principal Component Analysis (PCA), but AE gives higher efficiency due to the deep extraction of non-linear properties of the features. AE gives a representation of the output at each layer, giving multiple transformations at different dimensions of the input parameters. Data projections and visualizations using AE are more accurate than PCA and other dimensionality reduction techniques. In addition, outlier' identification is a by-product of any AE technique. PCA, LDA, and other standard ML feature extraction (data reduction methods) are used as they can be implemented easily. However, these methods are less capable in modeling the nonlinear structures of data, compared to the DL methods, especially in large datasets. The compressed representation of data with deeper AE networks can improve model performance [44].

One of the hybrid feature-extraction methods is proposed by merging Sparse AutoEncoders (SAE) and PCA to extract low-level features and is applied to various classifiers [45]. The results prove that the large numbers of nodes in hidden layers and deep information extraction from features are two critical parameters for achieving high

performance [46]. Traditional AEs fail to inspect relationships of data samples. They generate new features by minimizing only the reconstruction loss of the data. To resolve this problem, we minimize the reconstruction loss for data and relationships between data features.

We have investigated various AE networks that may fit into our problem. Among all, we chose SAEs, as it allows us to activate a selected number of nodes, which is the first requirement for this research challenge. We merge correlated features into one, where we have to activate and de-activate nodes and merge them by clustering into one. Besides, SAE consists of a single hidden layer, which can be deployed on any of the IoT device irrespective of its resource constraints. The loss function of the network is constructed by penalizing the activations within the layer [47]. SAE is efficient in encoding the input data by approximating minimum error, which guarantees less loss of information, and extraction of the best feature representation. From the literature [48], even with the simplest algorithm, it is possible to extract useful features and achieve higher performance by focusing on the hyper-parameter choices rather than on the complexity of the algorithm. We have incorporated these choices diligently after experimenting with various values on the SAE.

Data Pre-processing and Incremental Learning

Recently researchers have shown much interest in incremental learning algorithms [49], also referred to Transfer Learning [50], Online Learning [51], and FL [52] to adapt to the complex real environments. Existing supervised incremental learning algorithms are Learn++ [53], incremental SVM [54], and incremental Support Vector Machines (SVMs) that is proposed to incrementally learn from a reduced number of support

vectors [55]. Incremental learning based on user-provided labels or pairwise constraints is not suitable for dynamic data. Using streaming data, it is not always possible to label all the arriving data. If the streaming data is available at time-stamp t, the labeling of those samples either manually or automatically by any method may occur at time-stamp $t+1$, which is challenging and time-consuming. Some researchers have proposed semi-supervised algorithms to resolve this issue by using Bayesian learning, Subspace learning [56], Clustering [57], and Classification techniques [58]. Fitting unlabelled data based on an inappropriate model misleads the learning process and degrades the performance.

A distributed self-governing model is proposed where the decision is taken locally on the data collection devices, i.e., the positioning of the hierarchical processing layers on the IoT devices. Distributed processing mitigates the latency between the data transfer and decision making [59] as well as helps the resource-limited devices to meet the demands of the DL solutions for video/ image classification. This model is unsuitable to be applied for intrusion detection as the raw data should be shared among the IoT devices for decision making and thus it does not guarantee the privacy requirements.

Most of the incremental learning techniques proposed in literature assume that the data is labeled, which is not practical and makes the IDS incapable of dealing with new attacks [60][54]. Some of the incremental semi-supervised algorithms have been proposed to deal with unlabelled data [61]. Nevertheless, these algorithms do not achieve good performance during the testing phase because of the repetitive pre-processing, high computational demands, and high memory consumption, as the huge amount of data is transferred from IoT devices to cloud/edge platforms where the IDS is placed to provide various services. The streamed IoT data is big and heterogeneous with a mix of data

with known and unknown labels arriving chunk by chunk, which may overwhelm the classifier and lead to slow classification [62]. Therefore, the accuracy of incremental DL models can decrease. The response time, i.e., latency between the IoT data and the DL classification has to be synchronized to handle the vast amount of real-time traffic. The operational stability of the IoT IDS might also get compromised when there is a delay between the IoT domain and DL models detecting intrusions [63].

To sum it all up, the accuracy reduction is not the only problem with the existing incremental IDS algorithms. The processing time, memory consumption, and decision time are high when deployed in a real-time IoT network, where the traffic patterns are dynamic. Sometimes, the classifier is jammed or gives incorrect results. Only a classifier with short response time and high accuracy can ensure the security of the network. To meet the requirements of quick and correct response from the DL model, there has to be a shift from the traditional way of data transfer from the IoT devices.

Privacy-preserving Decentralized Models

In this section, we review some of the recent studies focusing on intrusion detection based on FL. Research indicates that IDS gained enormous attention and usage in every domain of IoT environments, right from smart homes, healthcare systems to Cyber-Physical Systems (CPS). Numerous IDS models have been developed based on DNNs. For example, Yang *et al.* [64] modeled a zone partitioning IDS to identify known and unknown malicious activities in CPS through various zones that have been compromised. Likewise, Yang *et al.* in 2020 [11] have designed an IDS based on Convolutional Neural Network (CNN) for SCADA networks. Among DL models, CNN is highly used to build strong IDS as the convolutional architecture analyzes the features closely to differentiate

minor changes. One of the challenges in improving accuracy of DL based intrusion detection models is optimal features that can be obtained by appropriate pre-processing techniques. Otoum *et al.* [65] have proposed an IDS using hybrid pre-processing techniques to select optimal features for training an efficient model. The model achieves high accuracy and is able to identify anomalous traffic patterns.

Most of these models are trained at a centralized entity, which is not only computationally expensive (time and processing) but also threatens the privacy of the data [14]. Though central models trained using ML or DL give higher accuracy, it overburdens the classifier with huge traffic from all over the IoT devices in the network. Besides, there is a request-response delay between the classification and the IoT data. Therefore, there is a shift towards distributed, decentralized, and similar approaches for IDS. Gajewski *et al.* have proposed a distributed IDS for smart homes [66]. The data of all IoT devices is collected on a home gateway (HG) of the Internet Service Provider (ISP) and the IDS is built in two levels in a distributed fashion. Level one is the IDS at the HG, which analyzes the traffic of all IoT devices and gives alerts while level two is the Network-IDS provided by the ISP, which re-analyzes any malicious alerts by the IDS at HGs.

Nevertheless, decentralized models proposed in literature transfer the original data to other devices, i.e., other IoT device, Edge node or cloud, which makes the data vulnerable to attacks and leaks. Recently, FL is adopted for IDS to address centralization and isolated data issues [67] [68]. FL is a collaborative training platform for building a shared global model. It is a distributed ML approach involving Edge computing that preserves local data privacy. In this process, each available client, i.e., IoT device trains a model locally on its private dataset and sends the model updates to a central server. The central server aggregates the parameters from all the clients and send back the updated

model [69]. The training process of an FL model is shown in Figure 3.1.

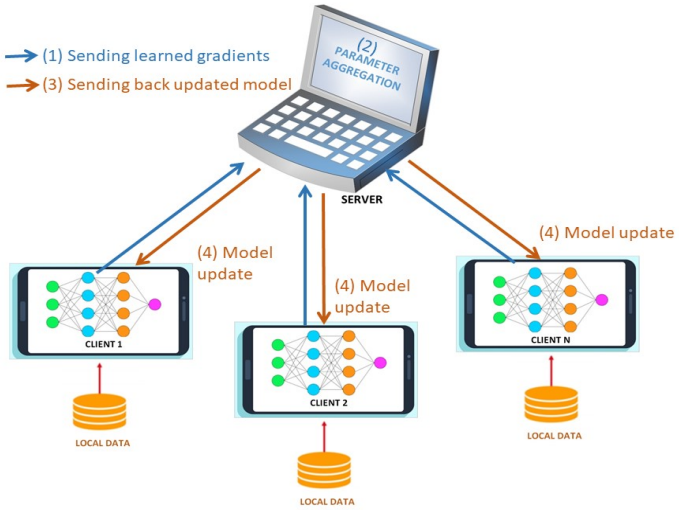

Figure 3.1: Federated Learning Approach.

FL trains the model locally, so it does not share the data with other devices ensuring data privacy. Nguyen *et al.* [70] have proposed a self-learning on-device IDS using FL for IoT devices. They modeled the IDS as an anomaly detection module that generates alerts about any communication deviation from the regular pattern in the IoT devices. Likewise, Yulin *et al.* [71] have designed IOTDefender, an IDS framework for 5G IoT devices using FL. The model is built in collaboration with all available IoT devices by transferring the parameters without privacy leakage. The model achieved 91% average accuracy with lesser FP rate than other unified DL models. Another intrusion detection model is built by [72] using Gated Recurrent Units (GRUs), which also uses regular federated architecture by sharing computed weights with the central server. IoT devices participate in model training using its local data and send the learned weights to the central server, which uses ensembles to aggregate the parameters. However, models trained using FL give poor performance and higher false alarm rates, if there is limited training data. Considering these arguments, we investigate intrusion detection models

built using GANs.

Data Imbalance Issues in Model Training

Seo *et al.* [73] have proposed a GAN-based driver safety system to reduce the false alarm rates in Vehicular Networks by augmenting the training data. In a fully decentralized way, Ferdowsi *et al.* [74] have designed an adversarial network to identify anomalies in IoT devices. This model aims to conceal the user's local data, i.e., IoT private data by using a single generator in the network and a discriminator network on each IoT device. However, the central generator collects the data distributions of all IoT devices in the network. Using a central generator for data analysis is a major bottleneck for scalability and request response. There are many GAN based intrusion detection models and federated intrusion detection models, but none of them is designed using GAN in a federated scenario for intrusion detection. We engage GAN in our model to synthesize training data to improve the model's accuracy by augmenting synthetic data. The data augmentation ensures less FP rates and better learning for minor class samples [75]. And, FL distributes the workload by enabling multi-party on-device learning. This ensures the privacy of data while allowing the data available in each device to be used in the training process.

CHAPTER 4: PARALLEL PRE-PROCESSING OF DATA ON IOT DEVICES FOR INTRUSION DETECTION

Any IDS involves two major steps: pre-processing and classification [76]. In ML, the pre-processing step extract or select features from the network traffic and send them to the classifier. Traditional ML algorithms can either select the features manually or using a predefined feature selection algorithm. Sometimes, the feature selection can cause the classification models to underfit or overfit, affecting the accuracy when dealing with big data. The traditional feature selection and classification algorithms do not perform well on big real-time data as it is heterogeneous with a huge noise and other irrelevant information. This needs efficient pre-processing to extract meaningful data for training. Another major reason where researchers fail to train models on real-time traffic is the unavailability of real-time traffic data sets. Besides, organizations do not share real data for training purposes due to privacy and security concerns. Moreover, using real-time data from different sources for training is vulnerable to manipulation attacks where the attacker injects malicious traffic patterns to corrupt the training model. Considering these problems, we propose a novel pre-processing technique using DL to extract less number of useful features to reduce the overhead on the classifier model. We use AEs with non-negativity constraints in order to help us extract high-level meaningful features. We pre-process the raw data on IoT devices using the AE network to ensure data privacy.

Since classical ML models suffer from a lack of scalability and low detection rate in large distributed IoT nodes. We chose DL network that can extract features automatically and provide adequate representation [77]. Moreover, the attack patterns change with time and newer features arrive in the network. Researchers have adopted incremental feature learning models to retrain the model in order to cope up with the emerging attacks

[78]. Likewise, we utilize our pre-processing technique using incremental learning in a distributed fashion to capture newer features. Once the newer features are captured, the classifier model is retrained to identify newer traffic patterns. The whole intrusion detection process is distributed in three networks designed for different tasks. When the raw data is sent directly to the classifier, various privacy concerns and confidentiality issues arise [79]. Data is vulnerable to be intercepted or reverse engineered by some adversaries, which may lead to privacy and security threats. We send the pre-processed data to the classifier, which is different from the raw input, to guarantee data privacy. Also, the risk of reverse-engineering the data decreases because the data processing is distributed over multiple distributed networks. In addition, we capture real-time IoT traffic to test our pre-processing and classification models.

System Model

We study a smart home consisting of various IoT devices, such as medical sensors, home appliances, and utilities as shown in Figure 4.1. To be more concrete, we consider the new medical applications that are becoming significant in our homes, such as early diagnosis and real-time patient monitoring. Real-time monitoring using vital signs and automated emergency response reduces the dependency of the patient-caregiver and decreases the healthcare costs. Different wearable IoT sensors interact with each other for communication using different protocols. The amount of traffic is large and the data is flowing continuously. The nature of the data is confidential and needs to be secured. This diverse environment provides a potential attacking surface and the attacker may manipulate the traffic patterns. If the manipulated patterns are not detected at the early stages, it may lead to the unavailability of the system. If an adversary compromises the

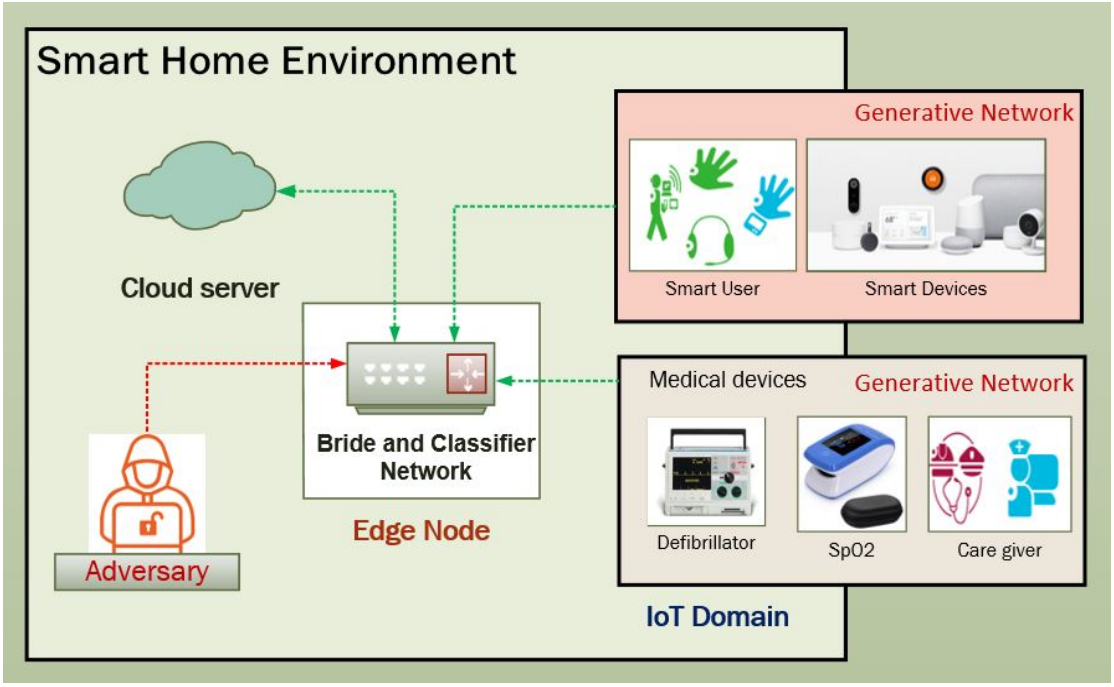

Figure 4.1: System Scenario.

smart home IoT health system network by attacking any one of the wearable IoT sensors, for instance, the vital signs can be changed threatening the life of a patient. An attacker may gain control of the whole network by compromising devices through performing different types of attacks. To protect the smart home against such malicious activities, we propose an IDS framework, where the data pre-processing is done on the available IoT devices and classification is done on the Edge device. To speed up the classification and minimize the input of the classification model, we pre-process the data on the IoT devices so that less data is sent to the remote classifier.

Notations

In this chapter, we denote F_i for input traffic, i.e., feature set from an IoT device, where each value is denoted as x_i, which is unprocessed. After pre-processing, the values are denoted as $\hat{x}_i$ and the whole set of pre-processed values from an IoT device

is denoted by G_i. The total number of IoT devices in the smart home health system is denoted by p, the number of input traffic, i.e., unprocessed is denoted by n and after processing the number is denoted by m. The final optimal set of features used for training is denoted by H and after every update, we denote it by H'. Based on these notations, C_i defines the nodes of the generative network with hidden layers as hl_i with weight matrix and biases denoted by W and b, respectively.

Distributed Architecture

The proposed distributed architecture is inspired by a method based on incremental semi-supervised learning on streaming data for video classification, which consists of three layers [80]. The first layer learns features from the incoming streaming data, the third layer regularizes the network by building similarity constraints. These two layers are connected by a bridge layer. Likewise, our model is segregated into three deep networks, namely, generative network, bridge network, and a classifier network. The tasks that are implemented in these networks are pre-processing, comparison and classification, respectively. The detection process is distributed over these networks right from pre-processing of the incoming streaming data to the classification results.

The proposed IDS identifies DoS attacks in IoT home network scenarios. The reason for selecting a DoS attack is because it generates huge traffic, which may shutdown or overwhelm the classifier before it could categorize the attack and notify the attack detection. It is considered to be the most possible attack in IoT devices due to their resource constraints, and especially for IoT health devices, it can be devastating if the device is shutdown. So, we have targeted DoS, and its categories of attacks. As our model is incremental, it can be retrained for many other attack identification tasks. In the

following subsections, we provide a detailed explanation of the tasks of each network.

Generative Network

In this network, huge traffic, i.e., either normal or malicious events are captured from different IoT devices. We build a layer of SAE on each IoT device to analyze traffic within the device. The collected raw data from each IoT device is recorded and pre-processed for unique feature identification. All AEs pre-process independently and unique features are extracted from the incoming traffic from each of the IoT devices simultaneously. Here, the unique features mean the features without redundancies and similarities. Similar features are merged based on the proposed heuristics. Once unique features are extracted on each IoT device, the useful features are sent to the bridge network for further analysis and recording of incremental features. The bridge network is separated from the classifier network, but both are deployed on one Edge device.

The networks and their tasks are illustrated in Figure 4.2, which shows three networks, generative, bridge, and classifier. The generative network takes the incoming traffic from the IoT devices as an input. Let us assume that the incoming traffic on each IoT device is $\{x_1, x_2, x_3, ...x_n\}$, we refer to it as input set of values F. The input set of values of each of the IoT device is referred to $\{F_1, F_2, F_3, ...F_p\}$, where '$p$' is the number of IoT devices in the smart home health system. Each input set of values are pre-processed to extract unique features. The input set of values $F_1 = \{x_1, x_2, x_3, ...x_n\}$ on IoT device 1 are converted to $G_1 = \{\hat{x_1}, \hat{x_2}, \hat{x_3}, ...\hat{x_m}\}$, which are the unique extracted features. Likewise, $\{G_1, G_2, ...G_p\}$ are all sent to the bridge network.

We propose a heuristic algorithm for feature extraction from the streaming IoT data and automatically compare features for duplicity and fuse it into informative deep

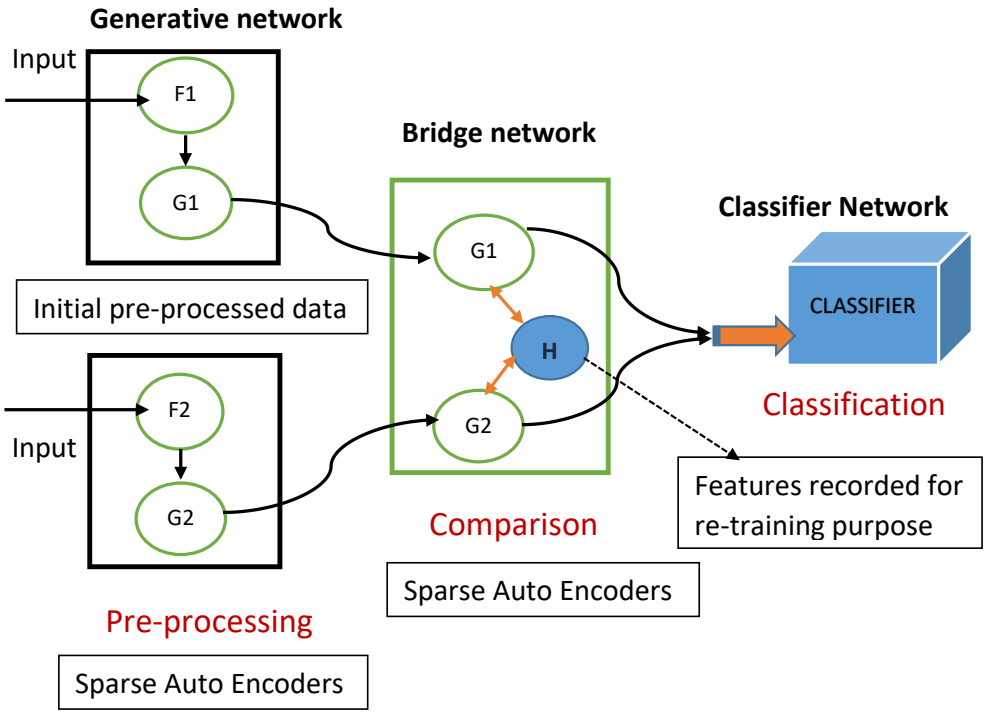

Figure 4.2: Distributed Architecture for Pre-processing and Incremental Learning.

features. The algorithm is elaborated in the next subsection in Algorithm 1.

Bridge Network

The bridge network receives unique feature sets $\{G_1, G_2, ...G_p\}$ from all IoT devices. It forwards those values to the classifier network for classification results. Apart from this, it analyzes the received feature sets by comparing them to the existing features on which our classifier model is trained. After which, it records any new feature and forms a new filter in the network. The working of this network is explained in Algorithm 2 in the next section. Bridge network construction is similar to the AE layers on the IoT devices. This network can be either placed on a sub-edge node such as IoT Hub or the same node as the classifier. In our case, we have placed it on the same device as

that of the classifier. From Figure 4.2, $\{G_1, G_2, ...G_p\}$ are sent to the classifier, where the classification results are given as Anomaly or Normal. Besides, $\{G_1, G_2, ...G_p\}$ are processed through the filters of the bridge network to know if any new features are recorded from the incoming traffic and is named as a new feature set 'H'. This is used for retraining purposes.

Classifier Network

Inspecting every packet is time-consuming and computationally expensive in streaming data to identify intrusions. We pre-process packets diligently to send only useful data to the classifier. We used a CNN model to detect the intrusions over the network, which has higher accuracy and bigger data handling capability [81]. CNN helps to reduce false alarms and thus unnecessary service visits of the network administrator to identify if it is a real attack. It has multiple levels of abstraction that discriminate features easily without overfitting the model. In this network, classification is executed with faster response time to alleviate the latency between IoT data pre-processing and IDS decision making. The classifier network gives results after processing the feature sets from the bridge network. The activations are connected to small regions of the neurons and not in a fully connected manner.

Detailed Design

The intrusion classification process is carried out in three phases in the distributed deep networks as shown in Figure 4.3 and explained below.

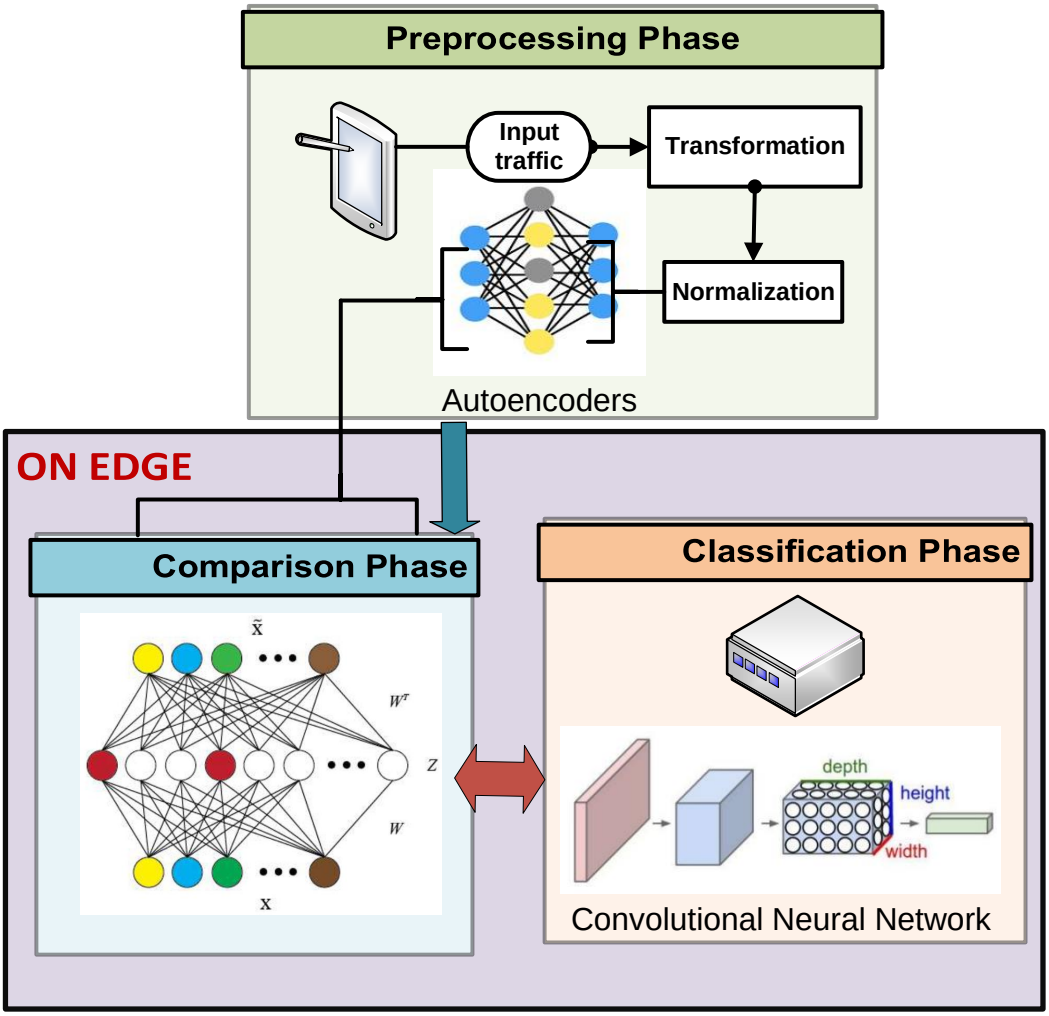

Figure 4.3: Complete Training Process Flow.

Generative Network: Pre-processing Phase

We train a deep SAE network on IoT devices for feature extraction using ReLU activation function. We used AEs as it detects rare events, i.e., outliers, and extracts high-level features, which are helpful to identify newer attacks. We perform the following steps in the pre-processing phase. Initially, the data is adjusted in one format and normalized so that we get improved results with our classifier. This is applied in both cases, Training and Testing.

Feature Transformation and Normalization

The incoming traffic is transformed into one format, for example, the text values are converted into numbers. Then, the transformed features are normalized using the Z-score function.

$$Z(i) = \frac{(v(i)\text{-}\mu)}{\sigma} \tag{4.1}$$

where μ is the mean of the n values for a given feature ($v(i), i\{\in 1, 2, 3, \ldots n\}$) and σ is the standard deviation.

Unique Feature Extraction and Fusion

Extracting the right features is often a complex and difficult task, which can be solved by using deep neural networks [82]. In this process, we use some heuristics that cluster and merge features by agglomerative clustering and reduction of hidden layer size. The most popular type of clustering techniques are agglomerative, hierarchical and K-means [83], among which hierarchical clustering is better except in terms of time complexity. In our technique, we aim to utilize the benefits of agglomerative clustering and also reduce the time it takes for this process. The two major tasks feature extraction and merging, are explained as follows:

- To extract a high-level unique feature set, we try to reduce the number of filters in the SAE network while preserving the sparsity. To eliminate redundant features from the network, we add a non-negative weights constraint to the network. The separation capability and sparsity of hidden layers is increased by utilizing ReLU activation function and a penalty factor. Traditional AEs fail to consider the relationships between the data samples. In our network, we consider reconstruction

loss of relationships by evaluating the correlation and similarity between data features and by filtering weak and trivial relationships, which helps in identifying similar features for the merging process.

- For feature fusion, the filters, i.e., nodes of the SAE network, that are identical, are merged using Agglomerative clustering. Average similarity threshold is calculated before the merging decision using minimal distance between two data features, and those are merged based on Agglomerative clustering. This deep neural network approximates the input vector with the minimum possible error. The minimal distance between two features is calculated using equation (4.2), where W is the weight of that feature.

$$\hat{M} = argmin\{x_i, x_{i+1}\} d(W_{x_i}, W_{x_{i+1}}) \tag{4.2}$$

Algorithm 1 illustrates the process of feature extraction and fusion.

We begin with the original filters, i.e., nodes of the SAE network. Let us assume it as C_i. Likewise, we have n number of nodes $\{C_1, C_2, C_3, ...C_n\}$. Each node on the SAE collects some incoming traffic, which we assume as $\{x_1, x_2, x_3, ...x_n\}$. At the beginning step, we check for correlation among the features and then initialize the weight and biases of the network (W, b). We model the SAE network in such a way that extracts high-level features with the least loss of information by minimizing the reconstruction error. We calculate the reconstruction loss every time by back-propagation until we reach the least value. Next, we add non-negative weight constraint to the network to eliminate redundant features, which also guarantees less reconstruction error, extraction of distinct features, and an increased sparsity. Then, we apply agglomerative clustering

Algorithm 1 Feature Extraction and Fusion

Input: $x_1, x_2, x_3, x_4, ..., x_n$

Output: Unique useful feature set = $\hat{x_1}, \hat{x_2}, \hat{x_3}, ...\hat{x_m}$

while corr $(x_i, x_{i+1}) \leq 90$ **do**
  initialize W and b
  initialize minRL
  while RL $\leq$ minRL **do**
    /**Begin Training**/
    W, b = Train_AE (dataset, W, b, iterations)
    /**Compute reconstruction loss **/
    RL = reconstruction loss (dataset, W, b)
    L1 + KL
    /**Backpropagation - Minimize RL **/
    minRL = RL
    /**using ReLU activation function **/
    Control penalty term
    Add non-negative weight constraint
    /** calculate threshold for similarity/ weak relationships **/
    Initialize threshold T
    T = argmin$\{x_i, x_{i+1}\}$ d $(W_{x_i}, W_{x_{i+1}})$
if function$(x_i, x_{i+1}) \leq$ T **then**
      Add $\hat{x_i}$& $\hat{x_{i+1}}$ to feature set
else
      Merge (x_i, x_{i+1}) as $\hat{x_i}$
end if

on the two most similar nodes C_i and C_j until the similarity is greater than the chosen threshold of distancing measure. We regularize the network to avoid over-fitting the data. The nodes C_i to C_n take input values from x_i to x_n on each of the IoT device and applies Algorithm 1 to convert those values into unique values $\hat{x_i}$ to $\hat{x_m}$. Each hidden layer computation is defined as $[hl_i(x) = f(w_i^T.X + b_i)]$. The ReLU activation function that is designed to obtain a better representation of the input features is represented as $[hl_i(X, W, b) = ReLU(w_i^T.X + b_i)]$. We incorporate the ReLU activation function that increases separation capability and also the penalty factor is controlled to increase the

sparsity of hidden layers. The following three components control the network, which are the major components in the objective function of the SAE:

1. Reconstruction error over the entire dataset that has to be minimized. The reconstruction error is formulated as below for SAEs.

   $L(W, b)$

$$= \frac{1}{m} \sum_{k=1}^{m} || \ \sigma(W^{(2)} \sigma(W^{(1)} x^{(k)} + b_x) + b_h) - x^{(k)} \ ||_2^2 \tag{4.3}$$

2. Regularizer to prevent overfitting by reducing the magnitude of the weights to make sure that they are close to zero and non-negative. We incorporate L1 Regularization in our loss function, which is shown as follows:

$$\mathcal{L}(x, \hat{x}) + \lambda \sum_{i} \mid a_i^{(h)} \mid \tag{4.4}$$

3. Kullback Leibler Divergence: It defines relative entropy, which is a standard measure of the difference between the two distributions used to regularize the encoder. We add this as an extra penalty term to increase sparsity between the nodes.

$$\mathcal{L}(x, \hat{x}) + \sum_{j} KL(\rho \mid \hat{\rho}_j) \tag{4.5}$$

   ρ is a sparsity parameter that denotes the average activation of a neuron over a collection of samples, which is calculated as follows:

$$\hat{\rho}_j = \frac{1}{m} \sum_{i} [a_i^{(h)}(x)] \tag{4.6}$$

   The sparsity parameter is used to make some of the hidden nodes inactive, which in turn increases the sparsity between the nodes of the hidden layer.

Bridge Network: Comparison Phase

The bridge network is made up of filters on which our model is trained using SAEs. The bridge network takes input from the generative network using all IoT devices and transfers the same input to the classifier network. At the same time, it compares that input with the filters of the AE network that we formed initially. If ever the incoming features from the generative network do not match any of these filters, then a new filter is added to this network. It means a new feature has been added to our feature set on which we can categorize attacks. After which, we use this new set of features for retraining. The output of generative network SAEs is fed as input to the AE of the bridge network. Assume that the model is trained initially on a set of features $\{x_1, x_2, x_3, ..., x_m\}$, then we form filters $\{f_1, f_2, ..., f_m\}$ based on these features on which the model is trained. Let us consider the features arriving from various IoT devices after extraction are $\{\hat{x_1}, \hat{x_2}, \hat{x_3}, \hat{x_4}, ..., \hat{x_n}\}$. The following algorithm explains the working of the bridge network.

In the testing phase, this algorithm takes input values from all IoT devices and the input values are checked by passing them through the filters of the SAE network. The FILTER function, which is called in Algorithm 2, describes the task of new filter addition in the bridge network. A minimum distance is calculated for each of the features based on each filter. If that minimum distance is satisfied, then the feature characteristics are similar to the filter. So, we do not consider that feature as new and the value returned to the algorithm is 0. We set the minimum distance by cosine distance measure, which is used to identify the similarities of feature characteristics by calculating the normalized inner product [80]. If ever the cosine distance is higher than

Algorithm 2 Incremental retraining module

Input: $x(1)_1, x(2)_2, x(3)_3, ..., x(n)_n$
Output: Decision: Yes / No

while corr $(\hat{x_i}, y_j) \leq 90$ **do**
procedure FILTER($\hat{x_1}, \hat{x_2}, \hat{x_3}, \hat{x_4}, ..., \hat{x_n}$)
    while i $\leq$ n & j $\leq$ m **do**
    if argmin($\hat{x_i}, f[j]$) **then**
        return 0
    else
        return 1
    end if
end procedure
    /**Check new feature**/
if (newFeature!=0) **then**
        Retrain = 1
else
        Retrain = 0
end if

the minimum value calculated, then it is considered as a new feature and added as a new filter to the network. That gives a new set of features, which are used for retraining purpose.

Classifier Network: Classification Module

In the training and testing process, we apply a distributed strategy with three different networks. When all the networks training is completed for the first time, the CNN model is fine-tuned with a small learning rate. We apply the drop-out before the output layer to avoid overfitting. We model the network traffic patterns as time-series data. The input structure of the first layer in our network depends on the number of inputs received. We convert the 1-dimensional feature vector into axb two-dimensional feature matrix. The network is composed of a 2D convolution layer, a 2D pooling layer and a fully connected

layer with a 3x3 matrix of the convolutional kernel.

The specifications of the CNN used in the experiment are as follows:

- The 1-dimensional input (x_m) is converted into a matrix axb, i.e., in the case of NSL-KDD, after the pre-processing phase, 24 features are converted into 3x8 matrix.

- The first convolutional layer consists of a kernel size = 3x3, step size = 1 and 32 convolutional kernels. Followed by a 2x2 pooling layer and ReLU activation function. After which, two fully connected layers and one drop out layer is used for regularization. The final output layer is built using softmax activation to give out two classes: anomaly or normal.

- The average classification loss is determined between the predicted label $\hat{y}$ and the actual label y by the cross-entropy function. The output label is a binary vector with softmax activation to get one label out.

Experimental Results

In this section, we discuss the performance of the proposed method in reducing the complexity of pre-processing and classification tasks. We also describe the datasets used to test our model. Then, we compare the proposed method with related existing approaches. Besides, we evaluate the model by varying the hyper-parameters. Initially, we focused on feature extraction and later, detection accuracies of the CNN classifier for different datasets. All experiments are performed on the devices acting as the generative, bridge, and classifier networks with the following configurations: Intel(r) Core i7 CPU @ 3.40 GHz and a 64 GB of RAM running 64-bit Windows connected in a home network

of various smart IoT devices. We have used standard datasets and also real-time data generated from the various devices connected in the smart home IoT health system. The data of each IoT device is pre-processed separately and then sent to the second network, i.e., bridge for comparison and recording the future incremental features. Then, the same data values are transferred to network three for the classification. The time duration is the time elapsed for various tasks, which is recorded in seconds and milliseconds. The time duration is measured for the pre-processing task and also, for the entire training of the deep neural network.

Dataset

1. KDD 99 CUP is the most widely used dataset for intrusion detection training and testing purposes. Although many criticize this dataset for having redundant records that result in biased classification, we have used this dataset for evaluating our model [84] to ensure that our pre-processing technique is efficient in eliminating redundancies and also, to record the time that our model takes for different datasets.

2. NSL-KDD dataset is proposed to overcome issues of KDD 99 [84] and is considered as a benchmark dataset for intrusion detection. Experimental results using NSL- KDD dataset demonstrate that our method achieves higher accuracy than the other incremental models proposed in the literature for DoS attack identification.

3. We collected the realistic traffic generated by smart IoT devices. We have collected 2 GB of data by passing normal traffic and also, launching the attacks. The data collected is a combination of benign and malicious traffic. The performed attacks

are DoS, Ping of the Death and Smurf. These attacks are launched one after the other to test if our incremental model is efficient in recording newer features from different types of attacks.

We evaluate three of the significant areas of our proposed methodology; pre-processing task, classification task and incremental learning.

Pre-processing Task

The first input vector of the AE is the incoming traffic from the real-time IoT networking or the standard datasets described above. For each dataset, we divided it into three parts: training, validation, and testing with 60%, 20%, 20% data samples in each part, respectively. Because our proposed model is distributed, we pre-process the input data on different AE networks, i.e., IoT devices in the training and testing phase. In the case of NSL-KDD dataset, we used 30,000 training examples for DoS attack detection with 41 features, which are consisted normal and DoS attack samples in a balanced ratio. The next step is to split this dataset into training, testing, and validation parts. After which, we divided the input, i.e., 41 features on three different AE pre-processing networks with 15, 15, and 11 features on each of the networks. Likewise, we split the feature sets of KDD 99 and the real-time IoT dataset on three different AE networks.

Network performance with different constraints

We plot the Root Mean Square Error (RMSE) of the pre-processing network, i.e., SAE network to record the loss of information after the initial task of learning features. The results given in Figure 4.4, 4.5 and 4.6 indicate that for change in each parameter

of the network the results are different. In these three figures, SAE represents normal SAE using sigmoid activation function while NSAE and T-P-NSAE represent SAE with ReLU activation by incorporating non-negativity, and threshold and penalty constraints in the network, respectively. We have tested RMSE score in a step by step procedure by applying each of our constraints that we used in Algorithm 1. It is understood that RMSE score without any constraints for each iteration is very high, whereas it is decreased when we included the non-negativity constraint and it is reached at its lowest that we could achieve by incorporating similarity threshold measure using Agglomerative clustering and penalty factor. With the three datasets, we achieved fewer scores after the application of all of the constraints discussed previously.

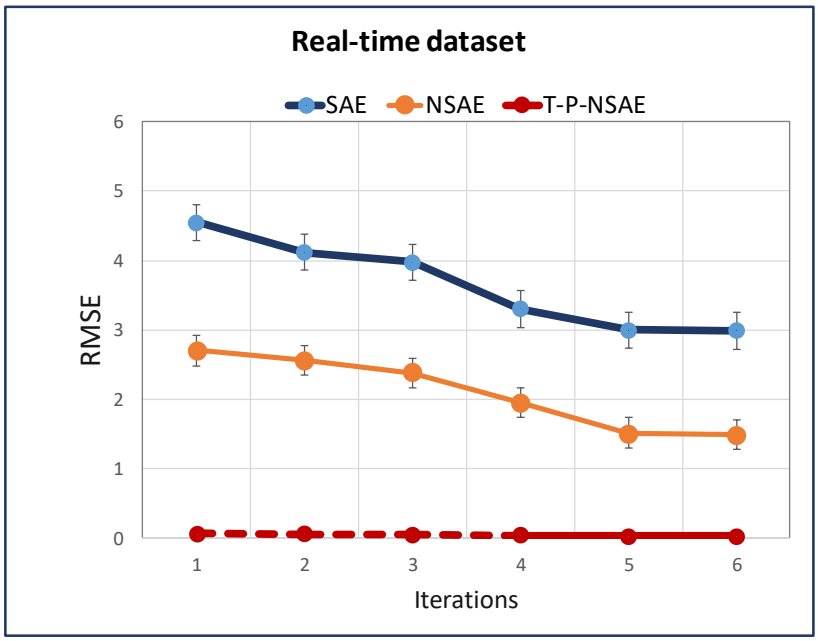

Figure 4.4: RMSE Scores of the Features on Real-time Dataset.

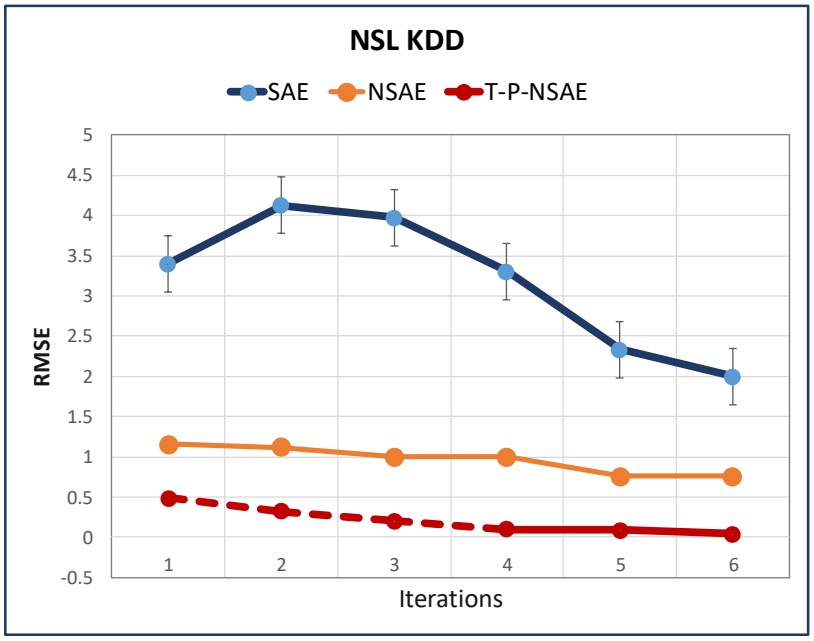

Figure 4.5: RMSE Scores of the Features on NSL-KDD Dataset.

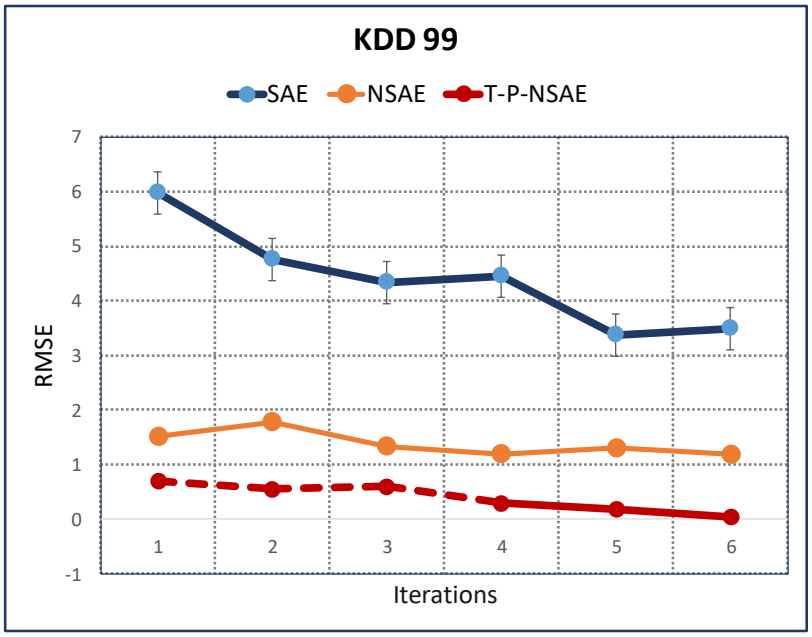

Figure 4.6: RMSE Scores of the Features on KDD 99 Dataset.

Reconstruction loss of network with a change of threshold

Now we look at the features learned from our pre-processing scheme on three different datasets, i.e., NSL-KDD, KDD99, and real-time IoT traffic. We plot reconstruction error versus similarity threshold to evaluate the performance of our model in extracting unique features. We begin the similarity threshold from 15 which is calculated from the heuristic from algorithm 1, and altered it until we reached the lowest reconstruction loss on the data. This comparison is a deciding factor for the value of the threshold that we determined for our algorithm using the Agglomerative clustering technique. Figure 4.7 shows the initial value of the reconstruction loss of the network that we obtained when we used 15 as the threshold. For simplicity, we converted the threshold value into natural numbers. Later, we tested by increasing and decreasing the threshold value to evaluate the performance of the network. In three cases, we increased the threshold gradually and fixed at 20 while training the model.

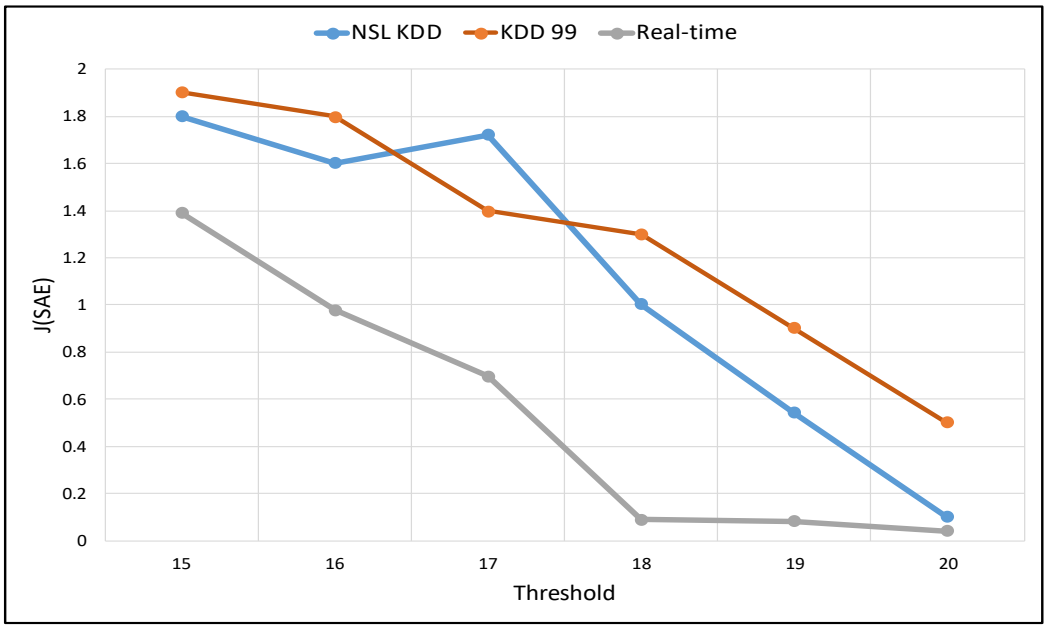

Figure 4.7: Sparse AutoEncoders Reconstruction Loss versus Threshold.

Table 4.1: Similarities of the Features Reconstructed.

NSL-KDD	$\leq$ Min(f_i, f_j)	KDD99	$\leq$ Min(f_i, f_j)	Real-time	$\leq$ Min(f_i, f_j)
f1, f2	✓	f1, f2	✓	f1, f2	✓
f1, f3	✓	f1, f3	✓	f1, f3	✓
f1, f4	✓	f1, f4	✓	f1, f4	✓
f2, f3	✓	f3, f2	✓	f4, f2	✓
f5, f6	✓	f9, f8	✓	f11, f12	✓
f9, f6	✓	f12, f14	×	f13, f17	×
f1, f9	✓	f13, f17	✓	f10, f12	✓

Features dissimilarity

We evaluated the quality of the extracted features by the distance measure. We chose some random features from different datasets and evaluated their similarity using the cosine distance. We calculate the cosine distance between features for evaluating the similarity of the features [80]. Table 4.1 shows that the distance between the reconstructed features is less than the minimal cosine distance defined for evaluating the uniqueness of the features. From the table, we infer that more than 90% of the features generated are unique with no similarities.

Classification Task

In the classification task, Accuracy, Precision, Recall, F1 score, and ROC curves of the model are employed as the evaluation metrics. Precision is the ratio of the number of correctly identified samples to the total number of identified samples. It evaluate how good our model is in identifying the input samples while the recall is the ratio of correctly identified samples to the actual correct samples. These two metrics are highly beneficial in benchmarking DL models. We have divided the dataset into three parts:

training, validation, and testing so that we validate the data to avoid overfitting. We have also regularized the CNN by incorporating L2 regularization and dropout layers. We evaluate the performance of our classifier model in the following way.

We plot the classification accuracy of our model in the training and testing phases. Figure 4.8 shows the accuracy and the full classification report is shown in Figure 4.9 obtained by applying our pre-processing technique and Convolutional Neural Network classifier. It can be seen that our proposed model performance is better compared to most of the attack detection DL models for all the three datasets, with the highest accuracy of 99% for NSL-KDD and KDD99, and in between 96% - 97% for real-time IoT traffic for DoS attack detection. The accuracy that we achieved on real-time traffic like this is because the incoming IoT raw data is imbalanced, which affects the classification results. Apart from this, the precision, recall, F1 and AUC scores are uniform with the accuracy results. Training and testing loss are another significant metrics that have to be checked to ensure that our model does not overfit the data and also, to confirm that it has acceptable biases and variances. For our model, we have obtained less value in the testing phase when compared to the training phase, so we conclude that our model does not overfit.

Incremental Learning Module

Once the data arrives from the first network, network two, i.e., bridge network records those data values and checks if it fits in the filters on which the bridge network is made. If any value does not match the filter characteristics based on the heuristic that we have explained in Algorithm 2, then that new value is formed as a new filter. The model is then retrained based on new and old values. To experimentally check the

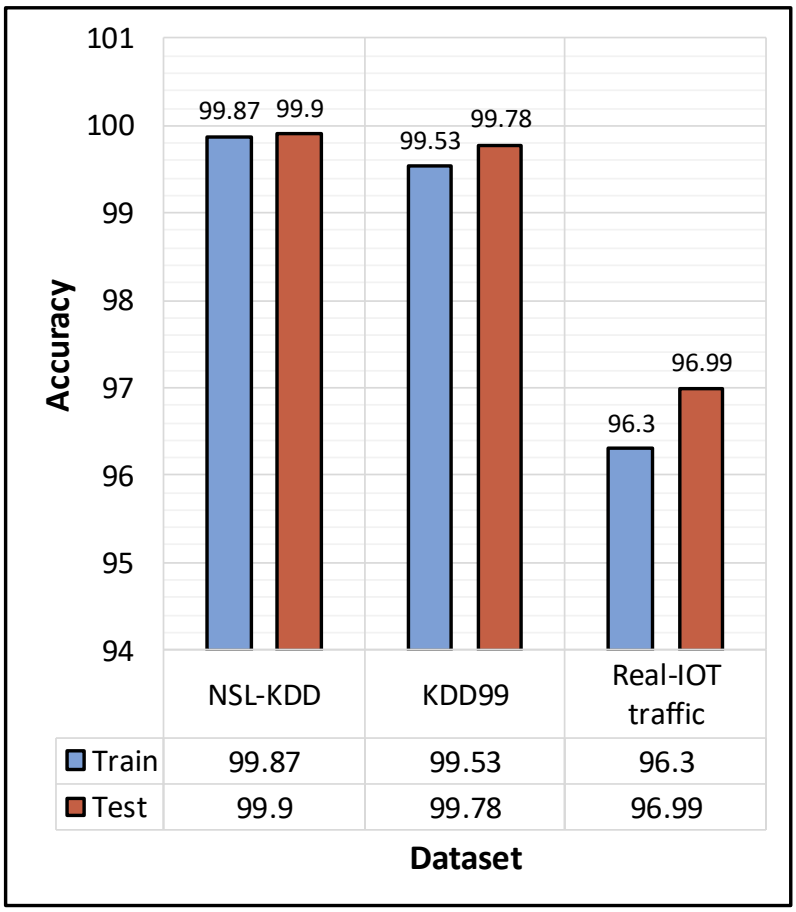

Figure 4.8: Accuracy of the Proposed Model.

heuristic proposed, we launched ping of the death attack in IoT smart home environment. The data collected during this attack is passed to the pre-processing networks. After the pre-processing phase and deep feature extraction by our technique, we finalized six unique features for that attack identification. In the next stage, we have launched Smurf, i.e., another category of DoS attack. Applying the same pre-processing technique, we found 11 features that are useful in this category of attack identification. When these features are passed to the bridge network, it filtered them into 9 features, which are useful in both attacks identification. The model accuracy and performance before and

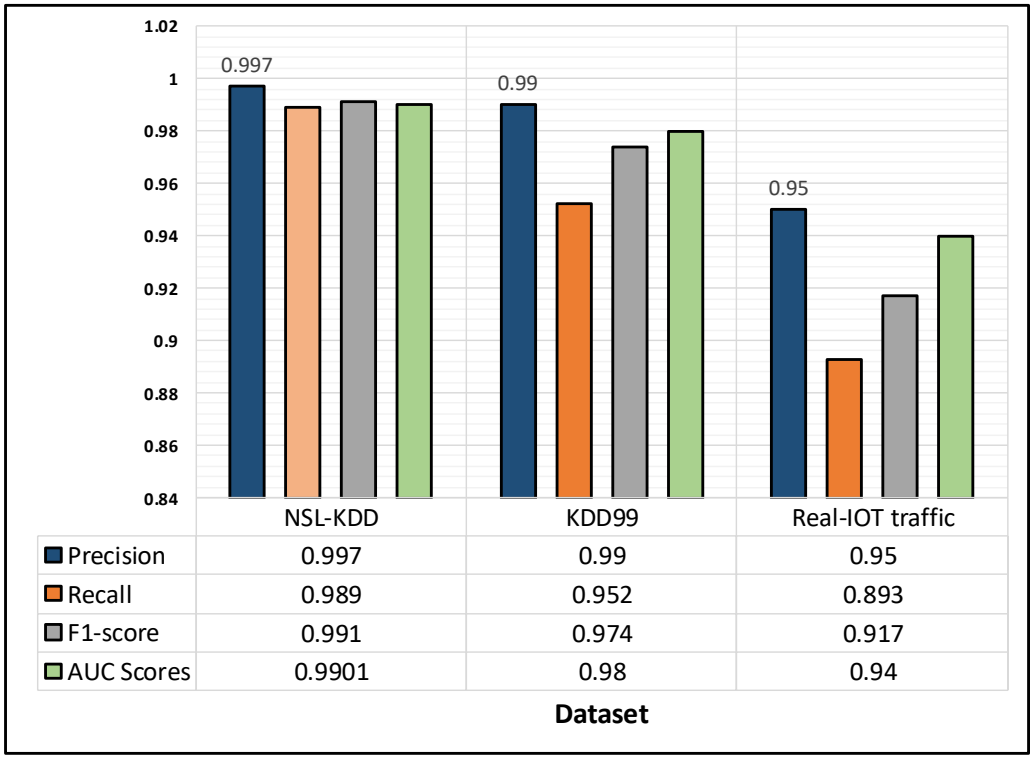

Figure 4.9: Full Classification Report of the Proposed Model.

after retraining with the newer features are shown in Figure 4.10.

Initially, the model that is trained using the NSL-KDD feature set is tested against the data collected for Ping of the Death attack. The model recorded 85% accuracy for classification of this attack while the pre-processing and comparison phase, i.e., on the bridge network gave a different set of unique features from the data collected during the ping of the death attack. This unique set of features are added as new FILTERS in the bridge network and the CNN classifier is retrained using the new set of final features. After the retraining, the accuracy of testing is increased to 94.8% for the same attack. Moreover, the data collected for another attack, Smurf, is also tested using the initial trained model on NSL-KDD, which gave 78% of accuracy and is increased to 95.5% after retraining with the features extracted and finalized through generative and bridge

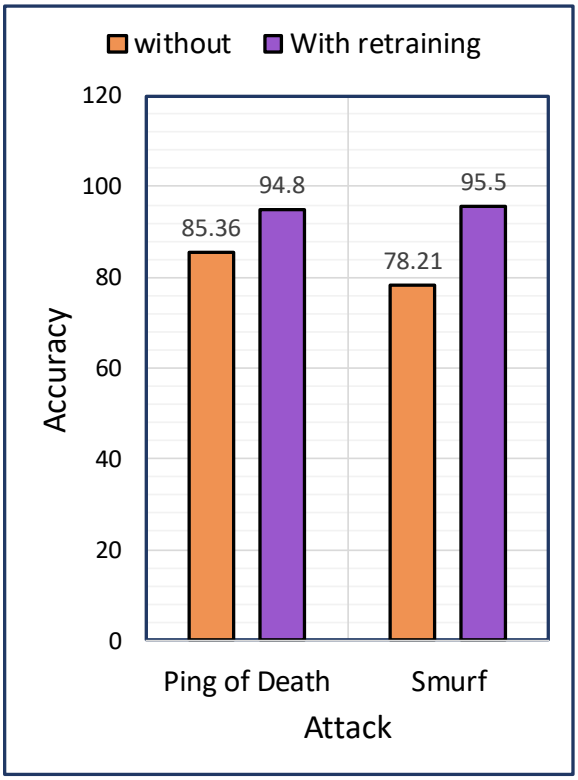

Figure 4.10: Accuracy before and after retraining for the Two Categories of Attacks.

networks. The increase in detection accuracy suggests that the incremental model for recording the newer features is appropriate for identifying the different types of newer attacks that the model is unaware of.

Time Complexity

After testing the performance of the improved feature extraction technique, we also measure the time taken by our technique in a centralized way and in a distributed way, i.e., in parallel by sharing the input vectors among three AE networks. We record the time during training and testing phases to ensure that it is suitable for real-time scenarios. The training complexity of AEs is determined by the complexity of the network structure

(the type of perceptrons). In our case, we use the Multi Layer Perceptron (MLP) with one hidden layer of k number of nodes, and thus the encoder and decoder have time complexity of O(m.(n+k)) [85] in the pre-processing phase, where n is the number of input nodes. We performed experiment on a dual-core CPU machine with four logical processors, at an average speed of 2-3 GHZ. In the NSL-KDD dataset, we used 30,000 training examples of 41 features divided into three sets. The results are shown in Figures 4.11 and 4.12 in the centralized and distributed way for pre-processing and classification tasks.

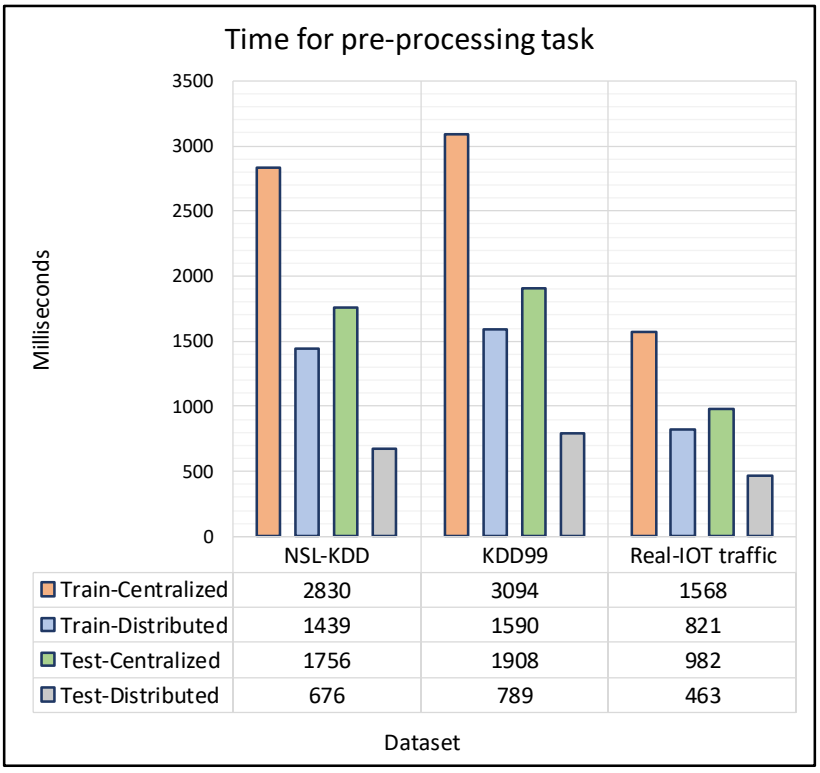

Figure 4.11: Time Taken for Pre-processing Task.

A single training epoch involving the pre-processing steps takes on average 1.4 seconds on NSL-KDD in the distributed setting, whereas for KDD99, it takes 1.59 seconds and for the real-time traffic, it takes 0.8 seconds. The input units of the AE are

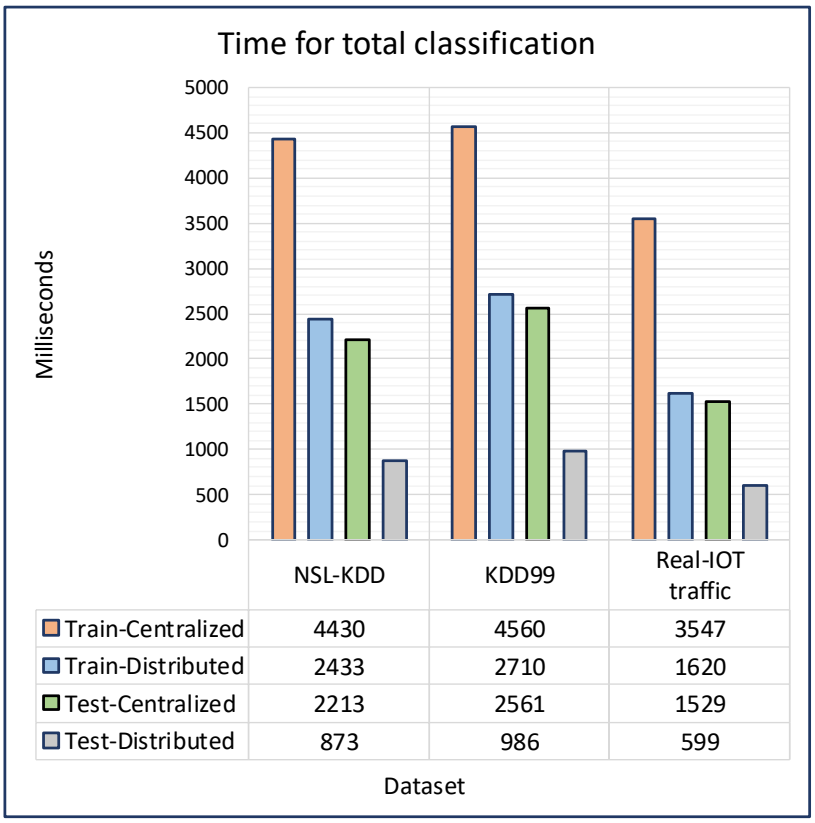

Figure 4.12: Time Taken for the Classification Task.

as per the input values of the dataset and real-time values collected from IoT devices. We also compared the performance against the pre-processing technique in the centralized way. The bar graph in Figure 4.11 illustrates that the time taken for training and testing phase by centralized, i.e., traditional pre-processing is higher for all the datasets, i.e., either standard or real-time IoT traffic. Confirming that the distributed/ parallel pre-processing method is more suitable as it takes less time and computations.

We record the time taken for the classification task in the training and testing phases. For the classification task, we tested the time elapsed for the whole processes in a centralized way and in a distributed way. Figure 4.12 shows that the distributed model of the pre-processing and classification tasks takes less time than the conventional

centralized way. Finally, we compared the time complexity of our model with those of the existing models. One such CNN-based model is developed using Accelerated DNN structure, which is trained on quad-core processors with 8 threads in a parallel mode, which makes the execution much faster. According to their proposed mechanism, their model gave three times faster performance by training the DNN model in parallel mode on a quad-core processor. In our case, we have used a dual-core CPU without any accelerator [86]. The time duration recorded suggests that the time taken for our model is less when compared to that model, using the same hardware resources. Sending properly pre-processed informative data to the classifier saves a lot of time and computations in the training and testing phases. The summary of the results is provided in Table 4.2

Table 4.2: Results Summary.

Dataset / Details	NSL-KDD	KDD99	Real-time data
Total samples	50,000	50,000	50,000
Training split	60%	60%	60%
Testing split	20%	20%	20%
Validation split	20%	20%	20%
Total number of features	41	41	23
No. of IoT devices	3	3	3
Features split	15, 15, 11	15, 15, 11	9, 8, 7
Extracted no. of features from each	11, 6, 7	7, 12, 6	3, 4, 2
Total no. of features extracted	24	25	9
Improvement in Time complexity (Train)	49%	48%	47%
Improvement in Time complexity (Test)	50%	48%	52%

CHAPTER 5: FEDGAN-IDS: PRIVACY-PRESERVING IDS USING FL AND GAN

The particularity of DL systems in iterative learning followed by back propagation on big data mandates parallel or distributed training to reduce the computational complexity and ensure data privacy. Most of the IDS models using DL face the problem of data transfer among the IoT devices or with the Edge node on which the IDS classifier is placed. Some advancements like "parameter server" approach considered training the neural networks on a shared dataset without data transfer [87]. The parameter server framework introduced by Google [88], utilizes one or more central servers managing multiple workers and their updated states for parallel processing. The worker networks train on their local share of data and communicate their weights to a central server. Parameter server framework involves shared data storage. At the beginning of each iteration, the worker machines download the data from the storage for training. Whereas in FL, the data resides on each worker and the central server does not keep track of any individual worker. Only the model gradients are sent from the workers to ensure privacy.

Similarly, FL trains DL models on a set of active clients [69] [89]. FL is similar to parameter server framework with a little distinction that clients undergo many local iterations, i.e., local training between each global update, i.e., interaction with the central node. Because of numerous iterations and updates, only one subset of the devices is selected at each round. Also, some of the workers may become inactive after some rounds. At the beginning of each round, the active clients synchronize the local model updates with the central server. FL models are trained locally by devices and only the model updates are shared with the central server [90]. However, it is challenging to build a good FL model with the limited amount of data in each device.

A recent successful discovery in ML is the various applications of GANs, particularly in generating synthetic data to increase the number of data samples and to fix under-sampling problems [91]. FL has been applied on discriminative models successfully, but there are various on-going investigations on the application of generative networks in FL. Although some of the efforts have been made in modeling GANs in federated settings [92] [93], much efforts are needed to model it for real scenarios. A typical GAN architecture consists of two tightly coupled DNNs, a generator and a discriminator. Nevertheless, it can be structured in a distributed fashion adopting a similar method applicable to regular DNNs. We model a GAN network in a federated setting, where local generators and discriminators are synchronized periodically by a central generator and discriminator with continuous improvement by exchanging gradients and model upgrades. We aim to augment the existing data with similar samples to increase the number of samples, especially from rare classes to fix data imbalance issues.

System Model

In the case of smart home e-health systems, IoT devices are the source of data. Hence, FL suits best as the on-device data is more pertinent and sensitive. In order to show the application of the proposed framework, we consider the following system model. Consider a smart IoT system scenario, such as a smart home, e-healthcare system, or an ambulance enabled with smart IoMT devices, as shown in Figure 5.1. Parts of the data from the IoT devices are sent to the Edge device or to the external network, cloud servers, for further processing and analysis. Any health data could be sent via the cloud to the medical practitioner or other participants.

The components of the system that we aim to secure are as follows:

1. *IoT devices or Clients*: The source of data generation, which needs to be protected against privacy and security threats.

2. *Edge Node*: The Edge Node is a mediator between the cloud and the IoT devices. It serves as a central entity for FL and needs protection to secure the model and also, data residing on the Edge node.

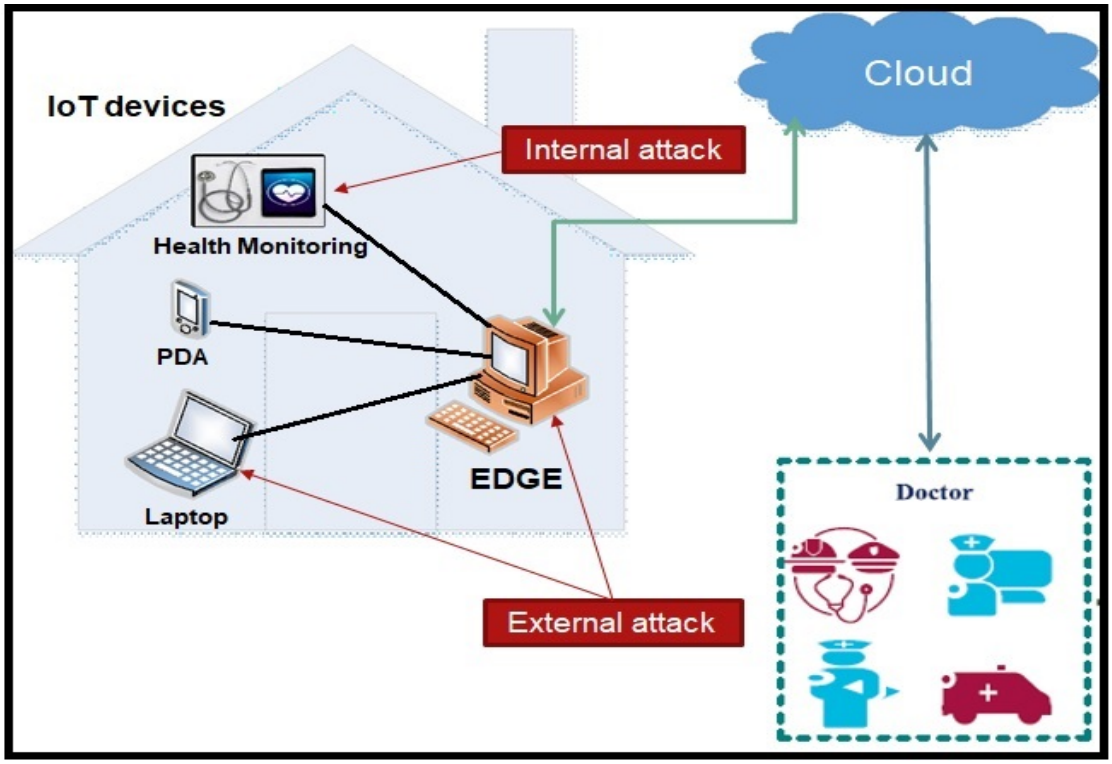

Figure 5.1: System Model.

The system is composed of a set of N IoT devices i.e., clients or worker machines, each consists of a local dataset B^n of size m and the number of features d that are transmitted through it. All the devices in that particular network follow a probability distribution $P_{data}(x)$, where x is Non-IID (Independent and Identically Distributed) i.e., time-series data, health records, temperature monitoring, or financial data. A set of data is said to be IID if the probability distribution of all random variables is same. In this

scenario, the probability distribution of the data on each IoT device may be different. The entire dataset of all active agents is denoted as $B = \cup_{n=1}^{N} B^n$, where B^n is the local dataset of each IoT device n. We aim to build an IDS to protect the IoT devices and Edge device from internal i.e., within the network and external attacks. Apart from attacks that target IoT devices, there are some attacks targeted towards the network. If we build a model by collecting all the data at one place, we endanger the privacy of the critical data on those devices. Besides, data centralization increases the communication overhead and the data can also be easily manipulated at one central entity. Thus, the local datasets in each worker should remain in place and should not be sent for training on other devices.

Proposed Framework

In this section, we first introduce distributed GAN setup for intrusion detection and adaption of FL. The complete architecture of our model is shown in Figure 5.2. Our approach is inspired by MDGAN [94] and FL-GAN [95] models, which are proposed for distributed datasets and privacy-preserving application, respectively.

Distributed IDS using GAN

We build a distributed IDS on IoT devices using GAN. Each IoT device has two networks, namely, generator and discriminator. The generator is a deep neural network that analyzes the local traffic of the IoT device and generates a similar traffic pattern. The synthetic data and original traffic of the IoT device are combined and provided to the discriminator network for training a classification model. The discriminator is a CNN that identifies malicious traffic patterns in the samples fed to it. Among DL models,

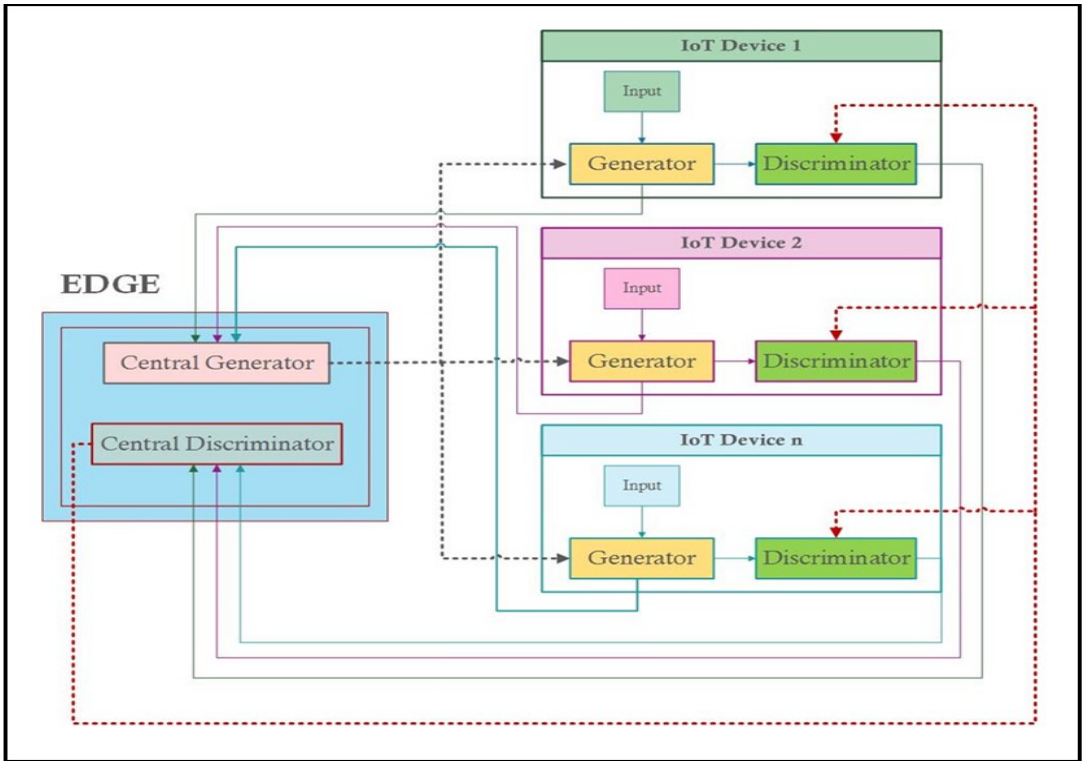

Figure 5.2: FEDGAN Architecture.

CNNs have gained huge success in the identification of malicious traffic in time-series data [96] [34]. Each IoT device builds its own local model on the discriminator network. The DNN networks give higher accuracy with a good amount of training data. With that motivation, we incorporate a generator network that generates similar traffic on each local worker and that augments the available data to train the discriminator used in the IDS. A central generator G^c and discriminator D^c is hosted on an Edge server in the smart home network. Likewise, local generators G^n and discriminators D^n are hosted by workers. The central generator and discriminator hold the initial parameters w and θ respectively G^c_w, D^c_θ. Each worker starts training its generator G^n with the initial parameter w_0, which is initially received from the central generator on its local dataset B^n. Similarly, the local discriminators D^n are trained with the initial parameter θ on its local dataset and data generated by the local generator $B_n + X_{G^n}$. The initial architecture

and parameters of the local generators and discriminators may be changed after their first local epoch. To improve the performance i.e., accuracy of the FL algorithm, we have used GAN.

Federated Learning Framework

In conventional FL, a subset of existing devices participate in FL. Each device downloads the central model and updates it based on its local data. Then, each device sends its updated model to the central model. The server aggregates these local models to construct an improved i.e., updated model. Likewise, the proposed federated structure aims to collectively build a distributed IDS. There are two central models: a generator and a discriminator. Initially, the central models select parameters: weights (w, θ), learning rate (α_g, α_d), batch size b, penalty coefficient (λ) and decay rate. Then, a ping message is sent to select the active devices in the network to send the parameters. Finally, the parameter sets of the generator and the discriminator are sent to initiate a training epoch.

There are numerous challenges for the efficient training of our model. The coupling between a generator and a discriminator requires organized strategies between the clients i.e., devices and dictates that the computational load on the devices be rational. Initially, each generator creates synthetic data by analyzing the probability distribution of the local traffic on the device. The discriminator takes the input from the local device and the local generator to train its network for intrusion detection. After completion of local iterations on two deep neural networks, the gradients are transferred to the central networks. The local model's parameters are aggregated by the central model and then the updated central model is shared with and adopted by the end devices. The

central generator and discriminator collect the gradients from all local G^n and D^n and aggregate them. The updated gradients are communicated back to the local networks on the devices. The local generators receive the updated gradients to build a new model in order to improve the quality of samples generated. The local discriminators improve its detection accuracy by the updated gradients from the central discriminator.

FEDGAN-IDS Algorithm

It is a Federated Generative Adversarial framework for training an IDS across distributed IoT devices to preserve the privacy of the Non-IID data. In our algorithm, a global communication round trains the local generators and discriminators on each IoT device that are periodically synchronized via a central generator and discriminator on the Edge that aggregates and transmits the new parameters to all local generators and discriminators.

Problem Formulation

We formulate the intrusion detection problem for multivariate time series as follows: In the first phase, the GAN model is trained based on the time-series dataset of each IoT device, X-input, and generates ”similar” samples X_G that "look like the dataset". We model the generator function as G_w: $R^l \rightarrow$ X. R and l are fixed. Likewise, the function representing the discriminator is D_θ: $X \rightarrow$ [0,1] where $D_\theta(x)$ defines the probability such that x is a normal sample while θ is the parameter of the discriminator. According to the parallel theory [97], the model passing through various clients by exchanging the model parameters does not expose the privacy of other clients. The local generators analyze the local traffic and generates similar traffic pattern, but only

the gradients of the local generators are transferred to the central generator. So, the output of the local generators on each client guarantees differential privacy as per the explanation provided in [95]. In every global communication round the parameters need to be exchanged among the clients, which gives the communication complexity for one round as $(|w| + |\theta|)$. The number of parameters passed also depends upon the number of generators and discriminators active in a communication round. So, the total communication complexity for all rounds is N.$(|w| + |\theta|)$. One major advantage of the proposed framework is the Edge device, which is placed nearby to the IoT devices that ensures better communication efficiency. The notations used in this chapter are given in the Table 5.1.

Table 5.1: Notations.

Notation	Description
D^n	Local discriminator
G^n	Local generator
D^c	Central discriminator
G^c	Central generator
N = $1, 2, 3..., n$	All devices or clients in the network
S_t = 1,2,3....,m	Available client/active set
w_t	Generator parameter at iteration t
θ_t	Discriminator parameter at iteration t
K	Synchronization interval
i..... I	Local training epochs
t = 0,1,2,.... E	Global training epochs
B^n	Local dataset on a device
ℓ_g^n	Loss of generator of client n
ℓ_d^n	Loss of discriminator of client n
E_g^n	Error feedback of client n on generator
E_d^n	Error feedback of client n on discriminator
b^n	batch size local
d^n	Size of dataset (feature set size) on a device
m^n	Number of samples in a dataset on a device
α_d	Learning rate of discriminator
α_g	Learning rate of generator
λ	Penalty Coefficient

A global communication round is composed of four phases:

- Phase 1 – Local Generator Training
- Phase 2 – Local Discriminator Training
- Phase 3 – Aggregate parameters at central model
- Phase 4 - Model Parameters Dissemination

The detailed explanation of each of these phases is provided in the below subsections.

Phase 1: Local Generator Training

The training begins from the local generators. Initially the local generator receives parameters (w_0 and α_g) from the central generator to begin training. Each local generator captures the real data distribution from the IoT device and generates similar traffic patterns. The number of local generators is equal to the number of available users i.e., IoT devices in the network. The generator is improved gradually with every local iteration. For training this network, we have used data samples from NSL-KDD dataset. The number of samples is denoted by *m* and the number of features is denoted by *d*. The training goes on for *I* local iterations with respect to the sample generation loss ℓ_g. The outcome of this network is a similar set of traffic pattern X_G. The operation of the local generator is explained in Algorithm 3. After completion of local iterations, the gradients of generator network and error feedback E_g^n is sent to the central generator. Error feedback of generator reports its amount of error in data generation on each device. The generated synthetic traffic is mixed up with the real data of IoT and fed to the local discriminator for training.

Phase 2: Local Discriminator Training

Step 2 in the process of training is the discriminator training for I local iterations. The operation of this step is given in Algorithm 4. The number of local discriminators is equal to the number of users i.e., IoT devices in the network. Local discriminator is trained on original traffic and generated traffic of the local generator to enhance the training process with enough number of samples. The data is pre-processed before training the intrusion

Algorithm 3 Local Generator

Input: Receive w_0 from the central generator. Receive B^n from local device. Set local iterations (I)

Output: Synthetic data from each generator

1: **procedure** LOCALGENERATOR(B^n, w_t)
2: **for** i $\longleftarrow$ 1 to I **do**
3: $Z_i \longleftarrow$GAUSSIAN NOISE (b)
4: $X_G^n \longleftarrow G^n(w_t, z) \mid \text{z } \epsilon\ z_i$
5: Calculate w_i^n

$$w_i^n \Longleftarrow Adam((\frac{1}{m}\sum_{i=1}^{m}\Delta_w L^i + n), w_i, \alpha_g) \tag{5.1}$$

6: Calculate E_g^n

$$E_g^n \Longleftarrow \frac{\partial B(X_G^n)}{\partial x_i}|x_i \epsilon (x_G^n) \tag{5.2}$$

7: Calculate L_g^n
8: **end procedure**
9: **Send** : w_i, E_g^n to the central generator

detection model. Local discriminator incorporates weights, biases, the penalty term and L2 regularization to overcome the problem of overfitting. The outcome of the network is the classification of the sample as attack or normal for a discriminator network designed for binary classification while for multiclass classification network it gives outcome as per the class of attack category. We also record the training and testing accuracy along with the loss of the discriminator ℓ_d. After the training process i.e., local iterations of each client, the discriminator computes an error feedback E_D. The computation of this function is explained in Algorithm 4. Discriminator error feedback reports its false positive and true-negative rate in anomaly detection process.

Algorithm 4 Local Discriminator

Input: Receive θ_0 from the central discriminator. Receive X_G^n from local generator of the device. B^n local dataset from the device. Set local iterations (I)

Output: Normal or Anomaly

1: **procedure** LocalDiscriminator(B^n, X_B^n, θ_t)

2: **for** i $\longleftarrow$ 1 to I **do**

3: D^n = X_G^n+B^n

4: Disc Learning (J_{disc}, D^n)

5: Update discriminator by ascending the stochastic gradient

$$\theta_i^n \Longleftarrow Adam(\Delta_\theta \frac{1}{m} \sum_{i=1}^{m} -D(G_\theta(z)), \theta, \alpha_d) \tag{5.3}$$

6: Calculate E_d^n

$$E_d^n \Longleftarrow \sum T.N + F.P \tag{5.4}$$

7: Calculate L_d^n

8: **end procedure**

9: **Send** : θ_i^n, E_d^n to the central discriminator

Phase 3: Central Model Update

Algorithm 5 demonstrates the training of this phase. The central generator and discriminator receive the parameters (w_t, θ_t) and the error feedback E_G E_D from all client models. The parameters received undergo federated averaging and gradients calculation to get the updated model parameters to be sent to all active clients. The two central networks broadcast w_{t+1}, θ_{t+1} to all clients. We perform E global iterations until a satisfactory accuracy is achieved on all discriminators. The central models try to minimize the error feedback of generators and discriminators at each global iteration.

Algorithm 5 Central GAN

Input: Initialize θ_0 for D^n. Initialize w_0 for G^n. Set Global iterations (E). Set the learning rate η at each iteration $\alpha(\eta)$ and b(η) synchronization interval K^t.

Output: Updated parameters

1: **for** t = 1,2,...,E **do**
2: Set of available clients: m
3: **for** m ϵ s_t **do**
4: $\rightarrow$ Local generator (w_t)
5: $\rightarrow$ Local generator (θ_t)
6: /** Each Client trains locally in parallel **/
7: **if** (t mod K) = 0 **then**
8: clients transfer gradients to central server

$$w_{t+1} \triangleq \sum_{n=1}^{m} P^n W_t^n \tag{5.5}$$

$$\theta_{t+1} \triangleq \sum_{n=1}^{m} P^n \theta_t^n \tag{5.6}$$

9: **end if**
10: Send the updated parameters to all clients; $w_t + 1$ and $\theta_t + 1$

Phase 4: Model Parameters Dissemination

The updated model parameters are sent back to all available clients to build their new models. In the testing phase of this framework, only the local discriminators are present for classifying the traffic of IoT devices and also the central discriminator on the central Edge device. After empirically determined *E* epochs i.e., global epochs, a final IDS is obtained.

Performance Evaluation

In this section, we provide a comparison of the performance of the IDS using FL with and without GAN in terms of test accuracy, training loss, recall, precision, F1-score,

AUC score and convergence rate. We also present the results recorded for binary and multiclass classification of the proposed IDS. The number of epochs for communication rounds i.e., global iterations is kept constant for accurate comparison. We perform all experiments using two standard datasets: NSL-KDD and KDD99. The ten labels of KDD99 dataset are categorized into four standard attack classes, DoS, Remote to local (R2L), User to root (U2R), and Probe. We perform all the necessary pre-processing steps for training the DL models. The experiments are conducted on an Intel(R) Core(TM) i7 - 10750H CPU @3 GHz predator machine with 4 GB NVIDIA GeForce RTX2060. For each training epoch i.e., global communication round, all the local model's parameters are transmitted to the central Edge node for aggregation and update.

Data Set

Various data sets like NSL-KDD, UNSW NB15 and KDDcup99 are available for testing the IDS. Each dataset has its own set of advantages and disadvantages. For instance, KDDcup99 contains redundant records and the categorization of malicious and benign packets is imprecise [98]. However, this dataset can be used to test the ability of the pre-processing techniques in eliminating the redundant data. Mostly, NSL-KDD data set is used for testing the IDS framework to identify its efficiency of detecting.

Federated Learning with and without GAN

In the first case, we model the IDS using FL without GAN. Each IoT device has only one Deep Neural Network and not the generator network. The single network is trained on the local data of the device. The data samples are divided randomly from NSL-KDD

dataset not considering any particular division of attack samples (DoS, PROBE, U2R, R2L). Whatever percentage of attack samples an IoT device gets, the same proportion of normal samples are fed to it to have a balanced local data for training. For example, IoT device 1 gets some samples of DOS and Probe, which constitutes 20% of the attack samples, and then 20% of normal samples are added to it. Likewise, IoT device 2 gets some samples of DOS and Probe, which constitutes 25% of attack samples from the dataset and then we add 25% normal samples to it. The local models are trained using the divided data on each IoT device. Once the models are built, the parameters of each model are transferred to the central node for aggregation from all IoT devices. Likewise, the accuracy of the model at each local epoch, each global iteration and the end of the training phase is recorded. As the data is Non-IID, we observe deviations in the results. The local iterations of each local model is considered 200, whereas the global iterations are chosen as per the performance of the models.

Likewise, we model the IDS using FL with GAN. Each IoT device has two Networks, generator and generator. The discriminator network is trained on the local data and generated data by generator network. The parameters of discriminator and generator from all IoT devices are transferred to the central generator and discriminator for aggregation. For this scenario, we also divide the NSL-KDD dataset randomly on the available devices. The following Figure 5.3 shows the accuracies of intrusion detection models using FL without and with GAN architecture. The accuracy is recorded on a IoT device after the final global update in both the scenarios.

The accuracy of both models increase gradually with the local epochs on the IoT device. However, the accuracy of the model without GAN network lies between 75% to 80%. While adding the GAN network helps to improve the accuracy at an earlier stage

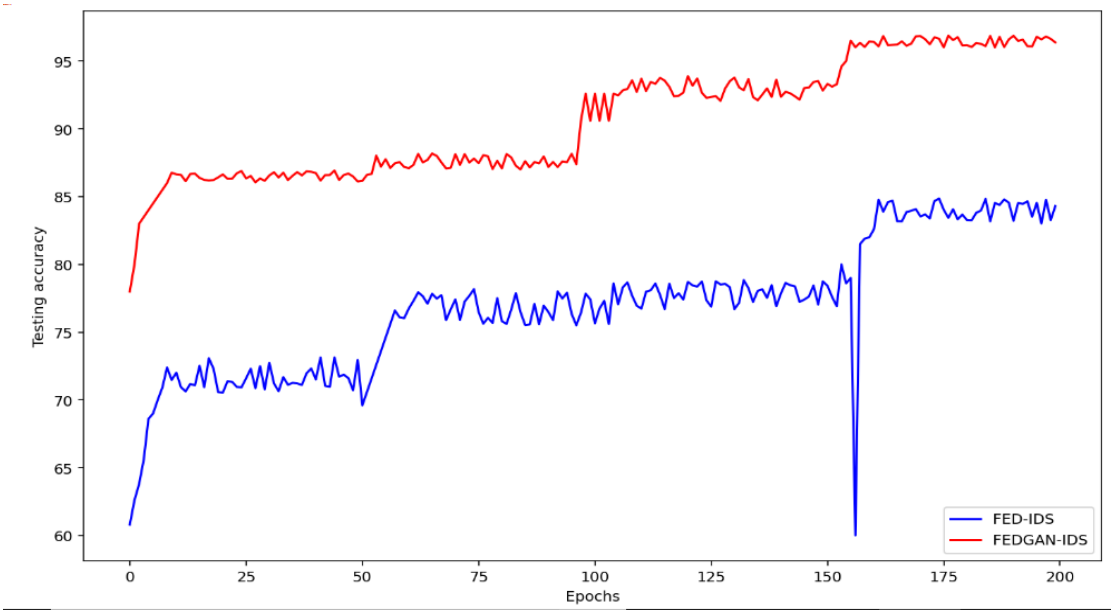

Figure 5.3: The Test Accuracy of a Local Model after Final Global Update of FED-IDS and FEDGAN-IDS.

in federated scenario. In FL with a GAN scenario, the model performance is enhanced in terms of accuracy and loss. Figure 5.4 shows the loss of the two IDS models. In FED-IDS case, the loss of the model moderately decreases and it reaches to the lowest value of 0.1. On the contrary, the loss is almost negligible i.e., 0.01 in FEDGAN-IDS case.

The two model's convergence is shown in Figure 5.5. After every iteration i.e., global epoch, the accuracy of both models is improved. At iteration 10, FED-IDS model achieves the accuracy of approximately 74% whereas FEDGAN-IDS model achieves more than 85%. The convergence state is reached 5 to 10 times faster in GAN case than the model without GAN. FED-IDS reaches to its maximum value 78% after iteration 25 and no significant enhancement is seen after that. With GAN network, the model reaches more than 95% of accuracy after 15 global epochs and the detection accuracy is 10% to 12% higher in Fed-GAN case. We deduce that adding GAN network to

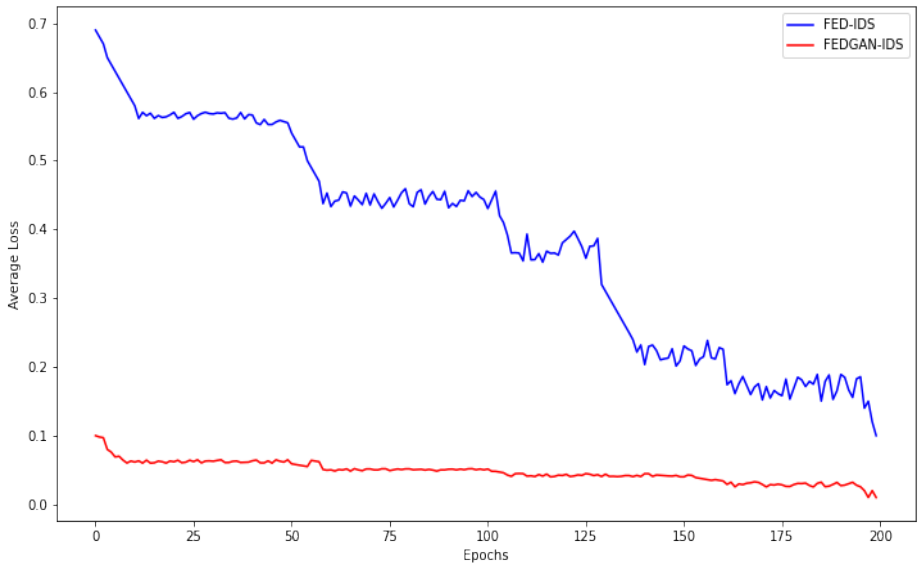

Figure 5.4: FED-IDS and FEDGAN-IDS Model Loss.

FED-IDS is promising for better performance of IDS model with FL.

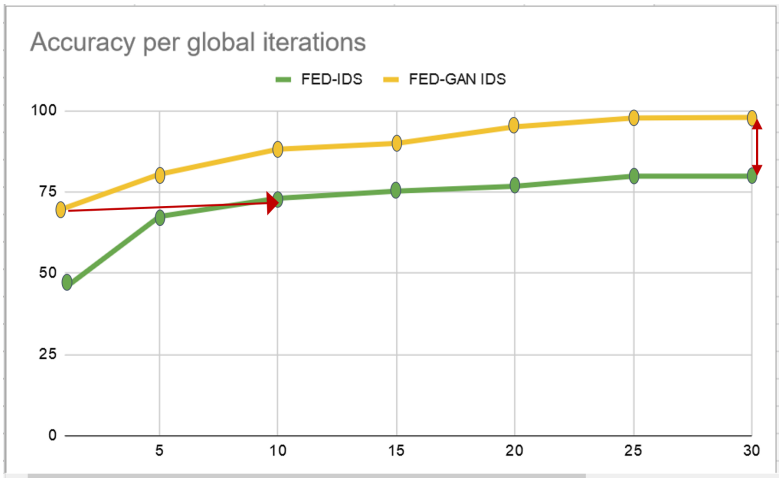

Figure 5.5: Model Convergence of Fed-IDS and FEDGAN-IDS.

We also show the performance evaluation of our FED-GAN classification model in terms of training and testing accuracy of local discriminators on its local dataset i.e., own data and synthetic data generated by local GAN. The training accuracy of the three

discriminator models is recorded at the Global iteration 3 i.e., after three global updates. The training and testing data splits are 70% and 30%. The discriminator models are trained successfully and each of the local models gives nearly equal accuracy. We record the training and testing accuracy of all local models during the local iterations i.e., 200 epochs and also during each global iteration. The test accuracy of three discriminators is evaluated on the 30% of the local dataset is shown in Figure 5.6. The test accuracy of three discriminators on different datasets is shown in Figure 5.7. At the beginning of the third global iteration, the test accuracy on the local discriminators is recorded from 0 to 200 local epochs. The test accuracy starts with a very low value and gradually increases to a satisfactory accuracy level with the local dataset. Whereas, on different datasets, there are many deviations in the results. The reason for testing the discriminator models performance on different datasets is to confirm that our model is not over-fitting. We incorporated the validation dataset while training to ensure the unbiased nature of the dataset.

Moreover, the accuracy of each local discriminator model is recorded while testing with the full dataset i.e., NSL-KDD for ten global iterations. The results are illustrated in Figures 5.8 and 5.9 for FED-IDS and FED-GAN-IDS models, respectively. FED-IDS models are trained on the local data of the IoT device and parameters are sent to the central model. After parameter aggregation, the updates are communicated back to all DNN models, which are retrained on the available local data of the IoT devices. After each global iteration, the test accuracy of the individual models is recorded to understand the performance of the model.

From the results, we observe that test accuracy improves with every update on all DNN models. FEDGAN-IDS model undergoes the same process but with two DNN

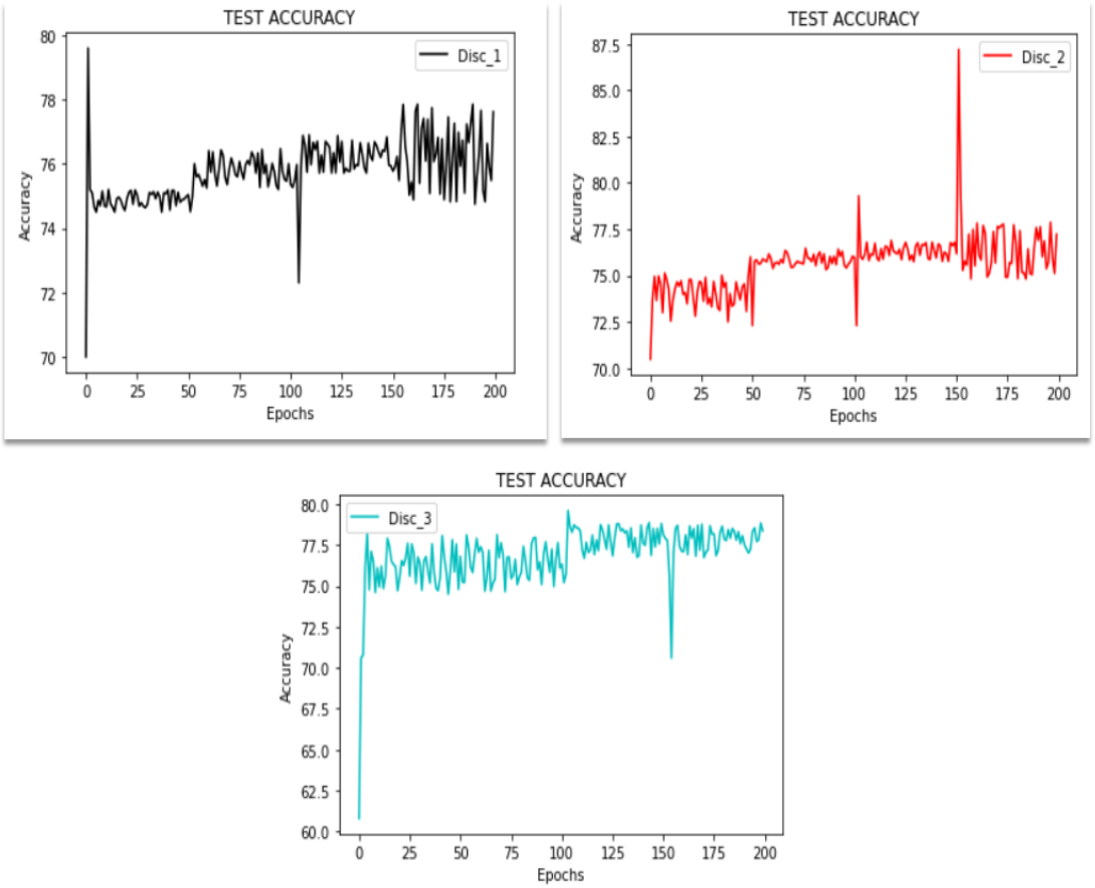

Figure 5.6: FED-GAN-IDS Test Accuracy on Local Discriminators at G_Iteration 3 with Local Test Data.

model updates at each global iteration. We are concerned with the attack detection accuracy of the IDS model, and hence we have recorded the performance of the discriminator models. These discriminator models are trained on local data of IoT devices and synthetic data generated by the generator network of that IoT device. The results signifies continuous improvement of the discriminator models at each iteration with better accuracy than the FED-IDS model. The summary of results for binary classification FEDGAN-IDS model using two datasets is shown in Table 5.2. The comparison between FED-IDS and FEDGAN-IDS in terms of all metrics is also shown in the table. The performance of FED-IDS is similar with two datasets, NSL-KDD and KDD99.

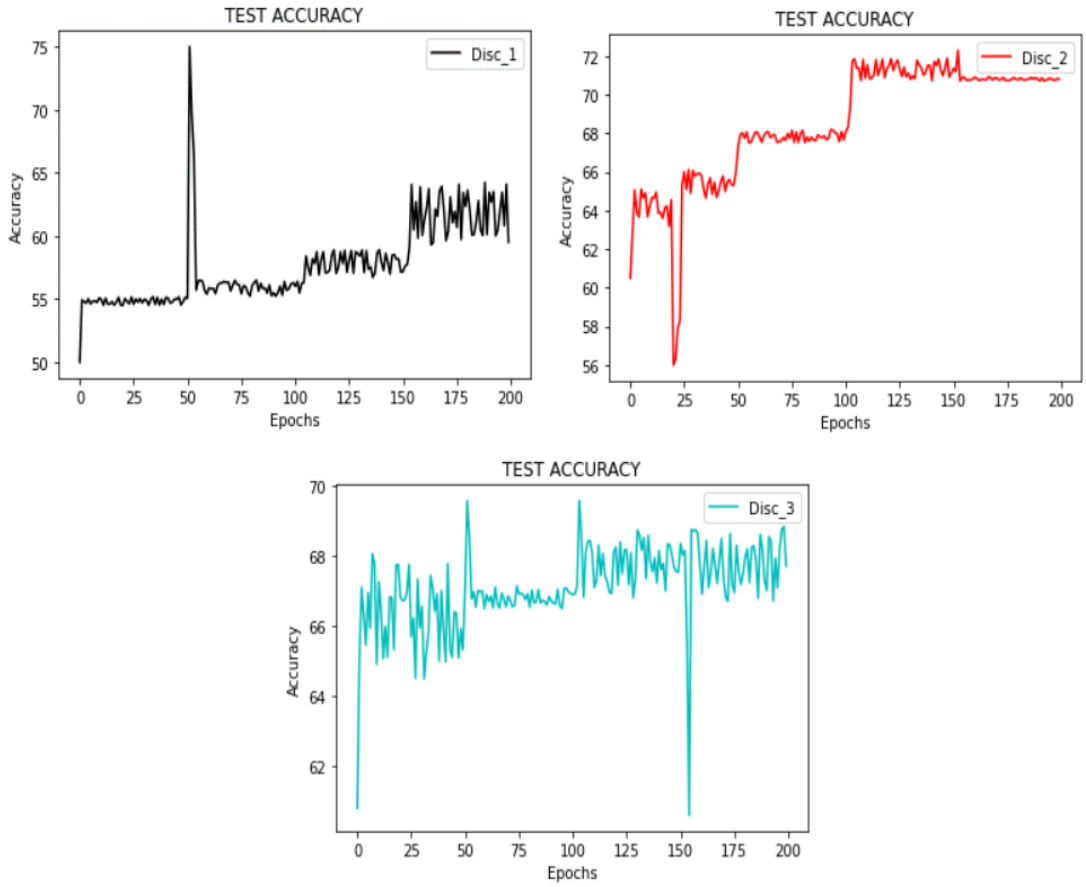

Figure 5.7: FED-GAN-IDS Test Accuracy on Local Discriminators at G_Iteration 3 with different Datasets.

FEDGAN-IDS Multiclass Classification

We also record the performance of multiclass FEDGAN-IDS model for different attacks. Figure 5.10 shows the performance recorded at each global iteration on discriminator1 modeled as multiclass. The selection of hyper-parameters played a principal role in training a multiclass FEDGAN-IDS model. The decay rate is selected as $1.0X10^{-4}$ and the penalty coefficient is 0.05. The generator on each IoT device augments the data for training the model. From the figure, we observe that the performance of GAN-based Federated multiclass classification IDS model is better than a non-GAN or standalone

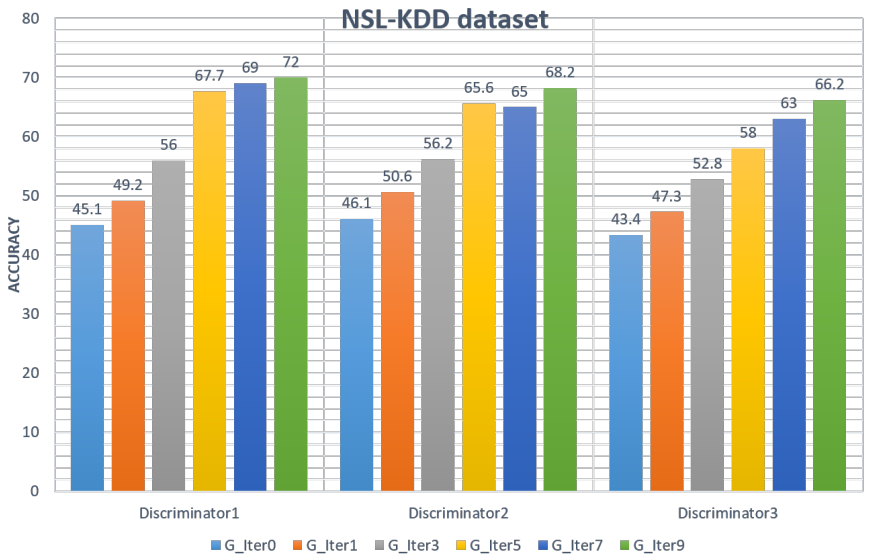

Figure 5.8: FED-IDS Binary Classification Test Accuracy at each Global Iteration.

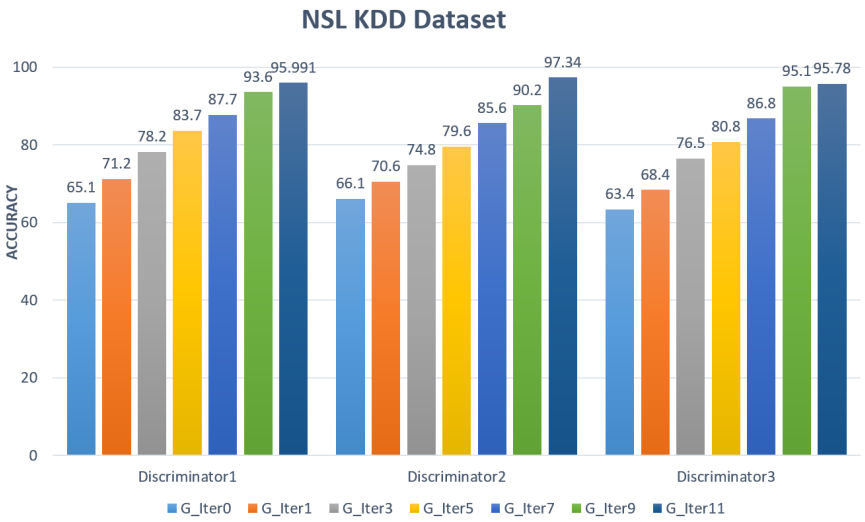

Figure 5.9: FEDGAN-IDS Binary Classification Test Accuracy at each Global Iteration.

IDS model. The accuracy of each attack class improves at each iteration. The detailed results after final global update are given in Tables 5.3 and 5.4 for the two datasets.

Table 5.2: Binary Classification.

Metrics	FED-IDS (NSL-KDD)	FEDGAN-IDS (NSL-KDD)	FED-IDS (KDD99)	FEDGAN-IDS (KDD99)
Accuracy	85.3	99.29	83.8	99.1
Precision	84.1	99.3	80.5	97.8
Recall	79.2	98.9	78.3	96.3
F1-Score	82.8	99	92.2	98.5
AUC-Score	78.4	99.01	79	98

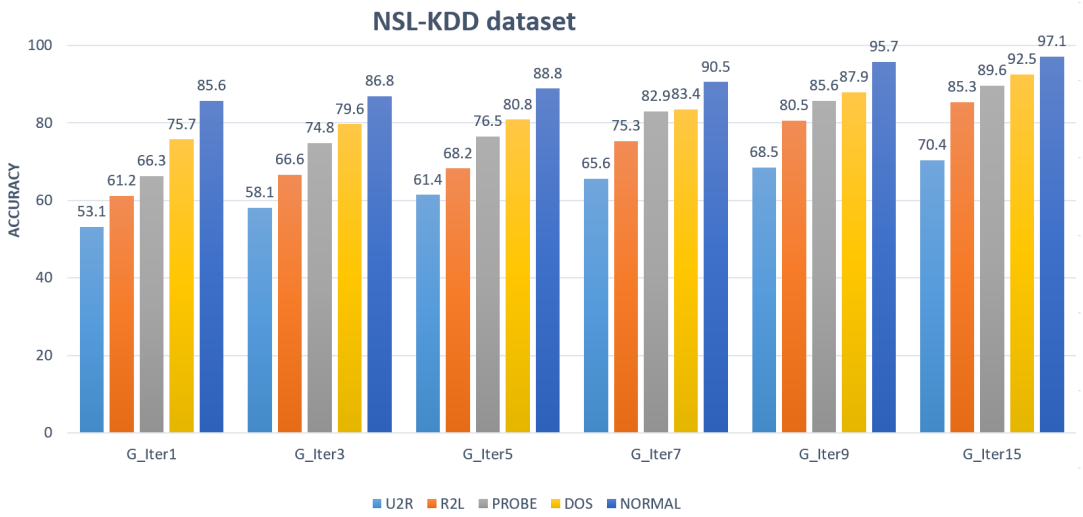

Figure 5.10: FEDGAN-IDS Multiclass classification.

Table 5.3: Multiclass Classification using NSL-KDD Dataset.

Metrics	U2R	R2L	PROBE	DoS	Normal
Accuracy	82	92.5	94.3	97	98.5
Precision	82.7	91.9	95	96.5	98.2
Recall	82.9	89.3	94.9	97.1	99
F1-Score	81.4	90.6	93	96.8	98
AUC-Score	82.3	90	90	96	97.5

Table 5.4: Multiclass Classification using KDD99 Dataset.

Metrics	U2R	R2L	PROBE	DoS	Normal
Accuracy	80.59	90	92	96.7	97.6
Precision	79.5	89.1	93.1	95.3	96
Recall	79.9	88.5	92.8	96	97.8
F1-Score	78.5	89.6	89	95.8	97.3
AUC-Score	77.4	87.8	87.6	96.9	96.4

CHAPTER 6: CONCLUSION

The number of smart devices, big streaming data, and the complexity of networks have made it difficult to secure the data and communications between IoT devices. Security vulnerabilities or random bugs in IoT devices can cause user dissatisfaction and various unpredicted outcomes. The open Internet architecture as well as the limited power, memory, computational capability and bandwidth resources make IoT devices potentially vulnerable for attacks. A notable amount of research proves that continuous network monitoring protects the network, so IDS is widely used defense mechanism. In this dissertation, we have investigated various existing IDS models for IoT devices to identify the current challenges. The major issues in building an effective IDS for IoT systems are resource constraints, heterogeneous big data, data isolation and privacy concerns. First, we have studied the resource constraints and big data issues, and proposed a distributed IDS with parallel pre-processing on IoT devices using incremental feature learning.

For tackling the big data, we have developed a heuristic pre-processing technique that greatly reduces the input to the classifier. For this purpose, a SAE network is placed on each IoT device to suit its resource restrictions. Each SAE pre-processes its local data and transfers it to a bridge network. The bridge network pre-processes the collected data from all IoT devices in the network and sends it to the final network for classification. The bridge network is also critical to capture any new features arriving in the network. Pre-processing on IoT devices reduces the input to the classifier and gives improved results. Incremental learning model records new features from the incoming traffic on a bridge network to retrain the model to identify newer attacks. Adding new features minimizes the objective function residuals. In addition, merging similar features avoids

overfitting and gives a compact representation of the features. The distributed setting of the proposed IDS guarantees less time and computations and categorizes attacks quickly. We showed that the model is able to achieve an optimal accuracy suitable for DoS attack detection using standard and real-time data set. This model can be utilized to retrain for different attacks by capturing the new features at the bridge network.

Second, we have studied the data isolation and imbalanced data issues, and proposed a GAN based IDS using FL that leverages the minor class samples. It is known that the accuracy of intrusion detection models is proportional to the amount of training data, especially for ML or DL trained models. However, each IoT device has a limited amount of data leading to weak separate models in Federated scenarios. Besides, it is not recommended to collect the data of IoT devices at one place for model training, particularly for e-health devices as the data is highly-sensitive containing information about the health conditions and other private patient information. Moreover, the traffic patterns are different on each smart device and can be used to train the IDS. If the data of all devices is utilized for training then the model performance improves.

Because of these significant obstacles, we have integrated two advanced technologies with complementary benefits: FL and GANs. A GAN network is modeled using FL, where each IoT device has a generator network and discriminator network. The generator on each IoT device analyzes the local data and generate similar traffic patterns to provide enough data samples for the local discriminator of that IoT device. After training of generator and discriminator networks, the gradients are transferred to the central generator and discriminator networks placed on the Edge, which aggregates and sends back the updated parameters. The two networks coordinate in building an efficient IDS model that ensures privacy of the data on IoT devices to some extent. The benefits of

this integration is highlighted with real-time scenarios and use-cases. After a series of evaluations and comparisons to various existing and proposed models, we proved that the FEDGAN-IDS architecture for IDS in the binary and multiclass classification scenarios is promising for present-day IoT networking. To the best of our knowledge, the application of GAN is used for the first time in the FL scenario for IDS. The model achieves 99% and 98% accuracy for binary and multiclass classifiers, respectively.

Scalability

Since IoT devices are hugely distributed with heterogeneous data and limited computational capability, applying FL in large scale IoT systems becomes difficult. The major bottleneck in training a successful FL model is the Non-IID nature and imbalanced data on IoT devices [99]. Considering these two significant problems, we chose a separate GAN network on each IoT device. Such that, it analyzes each device's traffic without the influence of other devices. The generator network learns independently based on the different traffic patterns of the IoT devices. Besides, the irregularity in data distribution and data samples tampers the FL training [99]. So, we augment the local data of the IoT device that ensures communication-efficient FL process. The data augmentation uniforms that the training process over the clients (IoT devices). The model that we have proposed is for smart-home scenarios which involves less number of IoT devices. However, the scalability of the model for large scale IoT systems can be tested by designing it using FL architectures that support huge number of clients. Such as "Flower framework" which is introduced to test and simulate FL process of large number of clients. As our proposed model attempts to solve Non-IID and unbalanced data issues in a smart home IoT system, it must resemble same properties in large scale IoT systems.

CHAPTER 7: FUTURE WORK

The proposed FEDGAN-IDS outperforms the classic centralized and distributed models. However, the federated approach leaves some backdoors for attackers to manipulate the training process or to compromise the model built [100] [101]. In the race for obtaining better models, FL fails to protect the architecture from inference and manipulation attacks [102] [103]. Attempts have been made to defend FL models against poisoning attacks using adversarial networks [92] [104]. If the training data is mixed up with some false data, then it computes different parameters. To some extent, it becomes difficult to launch membership inversion or inference attack to obtain the actual data set or a sample from the training set. In addition, the order of transfer of parameters and aggregation technique changes the model training. Nevertheless, FEDGAN architecture can be susceptible to various malicious attacks. In future, we plan to improve the following critical areas of the proposed model:

1. Though the local data on IoT devices is mixed with fake data generated by GAN network, it does not fully guarantee the differential privacy of the sensitive data on IoT devices. Future enhancements can to be made in order to guarantee the privacy of data.

2. Secure sharing of local gradients and global updates to ensure the integrity in order to protect the generator and discriminator networks training against interception attacks.

On the other hand, recent cyber attacks adversely cause physical damage beyond the cyberspace particularly in the area of smart health systems. Smart health devices such as IoT medical devices or Internet of Medical Things (IoMT), record and monitor human

vital signs for diagnosis purposes. These tiny devices are empowered with wireless connectivities to treat patients remotely [105]. Similarly, Implanted Medical devices (IMDs) are smart devices placed in the human body to monitor or treat various diseases in different organs or to enhance the poor functions of different body parts [106]. These remote monitoring devices generate a large volume of continuous data from the vital signs and other signals, e.g., EEG and ECG. This health data is confidential and sensitive to be attacked. Threats like eavesdropping, denial of service attacks or ransomware can cause monetary loss and sometimes loss of life in IoT systems [107] [108] [109]. Existing IoMT devices and IMDs in the market suffer from vulnerabilities, that if exploited can have dramatic consequences. We advocate that securing IoMT devices and IMDs is of utmost importance, as attacks not only expose sensitive patient data but can also lead to serious injuries and death.

The future directions for protecting these devices are as follows:

1. First we aim to customize FEDGAN-IDS for these devices. In designing FEDGAN-IDS for IoT devices, the computational and communication complexity is not considered, as IoT devices posses enough resources suitable to run a DL or FL model. However, IoMT and IMDs are empowered on batteries. Heavy computations and numerous communications of DL and FL models, respectively can drain the battery and shutdown the device. It is interesting to explore the possible ways of training of the FEDGAN-IDS intelligently with minimum communications and computations.

2. The IDS designed for these devices has to be light-weight since these devices are more resource constrained than the IoT devices. We investigate the possible ways

to optimize the proposed models to suit the deployment requirements on these devices.

3. Any tampering of the data on these devices can cause a serious issue, such as false diagnosis, delay in an emergency, and other health complications [110]. It is important to consider that the IDS model targeted for these devices has to be quick in malicious traffic identification along with higher accuracy.

PUBLICATIONS

- Tabasum, Aliya, Zeineb Safi, Wadha AlKhater and Abdullatif Shikfa. "Cybersecurity issues in implanted medical devices." In 2018 International Conference on Computer and Applications (ICCA), pp. 1-9. IEEE, 2018.
- Tabassum, Aliya, Aiman Erbad and Mohsen Guizani. "A survey on recent approaches in intrusion detection system in IoTs." In 2019 15th International Wireless Communications & Mobile Computing Conference (IWCMC), pp. 1190-1197. IEEE, 2019.
- Tabassum, Aliya and Wadha Lebda. "Security Framework for IoT Devices against Cyber-Attacks." International Conference on Internet of Things (CIoT 2019).
- Tabassum, Aliya, Aiman Erbad, Amr Mohamed and Mohsen Guizani. "Privacy-Preserving Distributed IDS Using Incremental Learning for IoT Health Systems." IEEE Access 9 (2021): 14271-14283.
- Tabassum, Aliya, Aiman Erbad, Wadha Lebda, Amr Mohamed and Mohsen Guizani. "FEDGAN-IDS: Privacy-Preserving IDS Using GAN and Federated Learning." Computer Communications | Journal | Elsevier (Submitted)

REFERENCES

[1] "Forecast economic impact of the internet of things (iot) in 2025 (in billion u.s. dollars),"

[2] E. Omanović-Mikličanin, M. Maksimović, and V. Vujović, "The future of healthcare: Nanomedicine and internet of nano things," *Folia Medica Facultatis Medicinae Universitatis Saraeviensis*, vol. 50, no. 1, pp. 23–28, 2015.

[3] P. V. Paul and R. Saraswathi, "The internet of things—a comprehensive survey," in *Computation of Power, Energy Information and Commuincation (ICCPEIC), 2017 International Conference on*, IEEE, 2017, pp. 421–426.

[4] A. Tabasum, Z. Safi, W. AlKhater, and A. Shikfa, "Cybersecurity issues in implanted medical devices," in *2018 International Conference on Computer and Applications (ICCA)*, IEEE, 2018, pp. 1–9.

[5] K. D. Kshirsagar, A. R. Attar, and P. V. Borade, "Iot based tire pressure monitoring system for vehicles," *Recent Trends in Control and Converter*, vol. 1, no. 1, pp. 17–21, 2018.

[6] M. Burhan, R. A. Rehman, B. Khan, and B.-S. Kim, "Iot elements, layered architectures and security issues: A comprehensive survey," *Sensors*, vol. 18, no. 9, p. 2796, 2018.

[7] C. Kolias, A. Stavrou, and J. Voas, "Securely making" things" right," *Computer*, vol. 48, no. 9, pp. 84–88, 2015.

[8] S. Hilton, "Dyn analysis summary of friday october 21 attack," *Dyn blog https://dyn. com/blog/dyn-analysis-summary-of-friday-october-21-attack*, 2016.

[9] B. B. Zarpelao, R. S. Miani, C. T. Kawakani, and S. C. de Alvarenga, "A survey of intrusion detection in internet of things," *Journal of Network and Computer Applications*, vol. 84, pp. 25–37, 2017.

[10] M. Plachkinova and C. Maurer, "Security breach at target," *Journal of Information Systems Education*, vol. 29, no. 1, pp. 11–20, 2018.

[11] B. Li, Y. Wu, J. Song, R. Lu, T. Li, and L. Zhao, "Deepfed: Federated deep learning for intrusion detection in industrial cyber–physical systems," *IEEE Transactions on Industrial Informatics*, vol. 17, no. 8, pp. 5615–5624, 2020.

[12] A. Riahi, E. Natalizio, Y. Challal, N. Mitton, and A. Iera, "A systemic and cognitive approach for iot security," in *Computing, Networking and Communications (ICNC), 2014 International Conference on*, IEEE, 2014, pp. 183–188.

[13] Z. M. Fadlullah, F. Tang, B. Mao, N. Kato, O. Akashi, T. Inoue, and K. Mizutani, "State-of-the-art deep learning: Evolving machine intelligence toward tomorrow's intelligent network traffic control systems," *IEEE Communications Surveys & Tutorials*, vol. 19, no. 4, pp. 2432–2455, 2017.

[14] H.-J. Liao, C.-H. R. Lin, Y.-C. Lin, and K.-Y. Tung, "Intrusion detection system: A comprehensive review," *Journal of Network and Computer Applications*, vol. 36, no. 1, pp. 16–24, 2013.

[15] S. Raza, L. Wallgren, and T. Voigt, "Svelte: Real-time intrusion detection in the internet of things," *Ad hoc networks*, vol. 11, no. 8, pp. 2661–2674, 2013.

[16] D. Oh, D. Kim, and W. W. Ro, "A malicious pattern detection engine for embedded security systems in the internet of things," *Sensors*, vol. 14, no. 12, pp. 24 188–24 211, 2014.

[17] T.-H. Lee, C.-H. Wen, L.-H. Chang, H.-S. Chiang, and M.-C. Hsieh, "A lightweight intrusion detection scheme based on energy consumption analysis in 6lowpan," in *Advanced Technologies, Embedded and Multimedia for Human-centric Computing*, Springer, 2014, pp. 1205–1213.

[18] V. Vaidya, "Dynamic signature inspection-based network intrusion detection," 2001, US Patent 6,279,113.

[19] P. Kasinathan, C. Pastrone, M. A. Spirito, and M. Vinkovits, "Denial-of-service detection in 6lowpan based internet of things," in *2013 IEEE 9th international conference on wireless and mobile computing, networking and communications (WiMob)*, IEEE, 2013, pp. 600–607.

[20] I. Butun, S. D. Morgera, and R. Sankar, "A survey of intrusion detection systems in wireless sensor networks," *IEEE communications surveys & tutorials*, vol. 16, no. 1, pp. 266–282, 2014.

[21] A. Mishra, K. Nadkarni, and A. Patcha, "Intrusion detection in wireless ad hoc networks," *IEEE wireless communications*, vol. 11, no. 1, pp. 48–60, 2004.

[22] A. Le, J. Loo, Y. Luo, and A. Lasebae, "Specification-based ids for securing rpl from topology attacks," in *2011 IFIP Wireless Days (WD)*, IEEE, 2011, pp. 1–3.

[23] A. Le, J. Loo, K. K. Chai, and M. Aiash, "A specification-based ids for detecting attacks on rpl-based network topology," *Information*, vol. 7, no. 2, p. 25, 2016.

[24] E. J. Cho, J. H. Kim, and C. S. Hong, "Attack model and detection scheme for botnet on 6lowpan," in *Asia-Pacific Network Operations and Management Symposium*, Springer, 2009, pp. 515–518.

[25] D. H. Summerville, K. M. Zach, and Y. Chen, "Ultra-lightweight deep packet anomaly detection for internet of things devices," in *Computing and Communications Conference (IPCCC), 2015 IEEE 34th International Performance*, IEEE, 2015, pp. 1–8.

[26] P. Pongle and G. Chavan, "Real time intrusion and wormhole attack detection in internet of things," *International Journal of Computer Applications*, vol. 121, no. 9, 2015.

[27] N. Chaabouni, M. Mosbah, A. Zemmari, C. Sauvignac, and P. Faruki, "Network intrusion detection for iot security based on learning techniques," *IEEE Communications Surveys & Tutorials*, 2019.

[28] C. Kolias, G. Kambourakis, A. Stavrou, and S. Gritzalis, "Intrusion detection in 802.11 networks: Empirical evaluation of threats and a public dataset," *IEEE Communications Surveys & Tutorials*, vol. 18, no. 1, pp. 184–208, 2016.

[29] M. S. Mahdavinejad, M. Rezvan, M. Barekatain, P. Adibi, P. Barnaghi, and A. P. Sheth, "Machine learning for internet of things data analysis: A survey," *Digital Communications and Networks*, vol. 4, no. 3, pp. 161–175, 2018.

[30] S. Rathore and J. H. Park, "Semi-supervised learning based distributed attack detection framework for iot," *Applied Soft Computing*, vol. 72, pp. 79–89, 2018.

[31] S. Chawla and G. Thamilarasu, "Security as a service: Real-time intrusion detection in internet of things," in *Proceedings of the Fifth Cybersecurity Symposium*, ACM, 2018, p. 12.

[32] T. N. Dinh and M. T. Thai, "Ai and blockchain: A disruptive integration," *Computer*, vol. 51, no. 9, pp. 48–53, 2018.

[33] N. A. Team, "Nebula ai (nbai)—decentralized ai blockchain whitepaper," 2018.

[34] A. Tabassum, A. Erbad, A. Mohamed, and M. Guizani, "Privacy-preserving distributed ids using incremental learning for iot health systems," *IEEE Access*, vol. 9, pp. 14 271–14 283, 2021. DOI: 10.1109/ACCESS.2021.3051530.

[35] M. Sakurada and T. Yairi, "Anomaly detection using autoencoders with nonlinear dimensionality reduction," in *Proceedings of the MLSDA 2014 2nd Workshop on Machine Learning for Sensory Data Analysis*, ACM, 2014, p. 4.

[36] A. A. Diro and N. Chilamkurti, "Distributed attack detection scheme using deep learning approach for internet of things," *Future Generation Computer Systems*, vol. 82, pp. 761–768, 2018.

[37] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, "Generative adversarial nets," *Advances in neural information processing systems*, vol. 27, 2014.

[38] L. Lan, L. You, Z. Zhang, Z. Fan, W. Zhao, N. Zeng, Y. Chen, and X. Zhou, "Generative adversarial networks and its applications in biomedical informatics," *Frontiers in Public Health*, vol. 8, p. 164, 2020, ISSN: 2296-2565. DOI: 10.3389/fpubh.2020.00164. [Online]. Available: https://www.frontiersin.org/article/10.3389/fpubh.2020.00164.

[39] I. Durugkar, I. Gemp, and S. Mahadevan, "Generative multi-adversarial networks," *arXiv preprint arXiv:1611.01673*, 2016.

[40] Q. Hoang, T. D. Nguyen, T. Le, and D. Phung, "Multi-generator generative adversarial nets," *arXiv preprint arXiv:1708.02556*, 2017.

[41] R. Bellman, "Dynamic programming," *New Jersey Google Scholar, Princeton University Press, Princeton, NJ, USA*, 1957.

[42] E. Lahner, M. Intraligi, M. Buscema, M. Centanni, L. Vannella, E. Grossi, and B. Annibale, "Artificial neural networks in the recognition of the presence of thyroid disease in patients with atrophic body gastritis," *World Journal of Gastroenterology: WJG*, vol. 14, no. 4, p. 563, 2008.

[43] C. Xing, L. Ma, and X. Yang, "Stacked denoise autoencoder based feature extraction and classification for hyperspectral images," *Journal of Sensors*, vol. 2016, 2016.

[44] J. An and S. Cho, "Variational autoencoder based anomaly detection using reconstruction probability," *Special Lecture on IE*, vol. 2, no. 1, pp. 1–18, 2015.

[45] K. Nandhakumar and D. S. Sukumaran, "A hybrid feature extraction method for network intrusion detection system," *IOSR Journal of Computer Engineering (IOSR-JCE)*, 4th ser., vol. 21, 3 2019.

[46] A. Coates, A. Ng, and H. Lee, "An analysis of single-layer networks in unsupervised feature learning," in *Proceedings of the fourteenth international conference on artificial intelligence and statistics*, 2011, pp. 215–223.

[47] C. Shi, B. Luo, S. He, K. Li, H. Liu, and B. Li, "Tool wear prediction via multidimensional stacked sparse autoencoders with feature fusion," *IEEE Transactions on Industrial Informatics*, vol. 16, no. 8, pp. 5150–5159, 2019.

[48] C. Zhang, X. Cheng, J. Liu, J. He, and G. Liu, "Deep sparse autoencoder for feature extraction and diagnosis of locomotive adhesion status," *Journal of Control Science and Engineering*, vol. 2018, 2018.

[49] D. A. Ross, J. Lim, R.-S. Lin, and M.-H. Yang, "Incremental learning for robust visual tracking," *International journal of computer vision*, vol. 77, no. 1-3, pp. 125–141, 2008.

[50] A. Argyriou, T. Evgeniou, and M. Pontil, "Multi-task feature learning," in *Advances in neural information processing systems*, 2007, pp. 41–48.

[51] G. Chechik, U. Shalit, V. Sharma, and S. Bengio, "An online algorithm for large scale image similarity learning," in *Advances in Neural Information Processing Systems*, 2009, pp. 306–314.

[52] D. Preuveneers, V. Rimmer, I. Tsingenopoulos, J. Spooren, W. Joosen, and E. Ilie-Zudor, "Chained anomaly detection models for federated learning: An intrusion detection case study," *Applied Sciences*, vol. 8, no. 12, p. 2663, 2018.

[53] R. Polikar, L. Upda, S. S. Upda, and V. Honavar, "Learn++: An incremental learning algorithm for supervised neural networks," *IEEE transactions on systems, man, and cybernetics, part C (applications and reviews)*, vol. 31, no. 4, pp. 497–508, 2001.

[54] Y. Yi, J. Wu, and W. Xu, "Incremental svm based on reserved set for network intrusion detection," *Expert Systems with Applications*, vol. 38, no. 6, pp. 7698–7707, 2011.

[55] A. Bordes, S. Ertekin, J. Weston, and L. Bottou, "Fast kernel classifiers with online and active learning," *Journal of Machine Learning Research*, vol. 6, no. Sep, pp. 1579–1619, 2005.

[56] J. Li, G. Han, J. Wen, and X. Gao, "Robust tensor subspace learning for anomaly detection," *International Journal of Machine Learning and Cybernetics*, vol. 2, no. 2, pp. 89–98, 2011.

[57] D. Kulić, W. Takano, and Y. Nakamura, "Incremental learning, clustering and hierarchy formation of whole body motion patterns using adaptive hidden markov chains," *The International Journal of Robotics Research*, vol. 27, no. 7, pp. 761–784, 2008.

[58] W.-F. Hsiao and T.-M. Chang, "An incremental cluster-based approach to spam filtering," *Expert Systems with Applications*, vol. 34, no. 3, pp. 1599–1608, 2008.

[59] C. Alippi and M. Roveri, "The (not) far-away path to smart cyber-physical systems: An information-centric framework," *Computer*, vol. 50, no. 4, pp. 38–47, 2017.

[60] Y. Wang, X. Fan, Z. Luo, T. Wang, M. Min, and J. Luo, "Fast online incremental learning on mixture streaming data," in *Thirty-First AAAI Conference on Artificial Intelligence*, 2017.

[61] F. Noorbehbahani, A. Fanian, R. Mousavi, and H. Hasannejad, "An incremental intrusion detection system using a new semi-supervised stream classification method," *International Journal of Communication Systems*, vol. 30, no. 4, p. 3002, 2017.

[62] M. H. Bhuyan, D. K. Bhattacharyya, and J. K. Kalita, "Survey on incremental approaches for network anomaly detection," *arXiv preprint arXiv:1211.4493*, 2012.

[63] F. Lin, Y. Zhou, X. An, I. You, and K.-K. R. Choo, "Fair resource allocation in an intrusion-detection system for edge computing: Ensuring the security of internet of things devices," *IEEE Consumer Electronics Magazine*, vol. 7, no. 6, pp. 45–50, 2018.

[64] J. Yang, C. Zhou, S. Yang, H. Xu, and B. Hu, "Anomaly detection based on zone partition for security protection of industrial cyber-physical systems," *IEEE Transactions on Industrial Electronics*, vol. 65, no. 5, pp. 4257–4267, 2017.

[65] Y. Otoum, D. Liu, and A. Nayak, "Dl-ids: A deep learning–based intrusion detection framework for securing iot," *Transactions on Emerging Telecommunications Technologies*, e3803, 2019.

[66] M. Gajewski, J. M. Batalla, G. Mastorakis, and C. X. Mavromoustakis, "A distributed ids architecture model for smart home systems," *Cluster Computing*, vol. 22, no. 1, pp. 1739–1749, 2019.

[67] H. Saadat, A. Aboumadi, A. Mohamed, A. Erbad, and M. Guizani, "Hierarchical federated learning for collaborative ids in iot applications," in *2021 10th Mediterranean Conference on Embedded Computing (MECO)*, IEEE, 2021, pp. 1–6.

[68] S. A. Rahman, H. Tout, C. Talhi, and A. Mourad, "Internet of things intrusion detection: Centralized, on-device, or federated learning?" *IEEE Network*, vol. 34, no. 6, pp. 310–317, 2020.

[69] Q. Yang, Y. Liu, Y. Cheng, Y. Kang, T. Chen, and H. Yu, "Federated learning," *Synthesis Lectures on Artificial Intelligence and Machine Learning*, vol. 13, no. 3, pp. 1–207, 2019.

[70] T. D. Nguyen, S. Marchal, M. Miettinen, H. Fereidooni, N. Asokan, and A.-R. Sadeghi, "Dïot: A federated self-learning anomaly detection system for iot," in *2019 IEEE 39th International Conference on Distributed Computing Systems (ICDCS)*, IEEE, 2019, pp. 756–767.

[71] Y. Fan, Y. Li, M. Zhan, H. Cui, and Y. Zhang, "Iotdefender: A federated transfer learning intrusion detection framework for 5g iot," in *2020 IEEE 14th International Conference on Big Data Science and Engineering (BigDataSE)*, IEEE, 2020, pp. 88–95.

[72] V. Mothukuri, P. Khare, R. M. Parizi, S. Pouriyeh, A. Dehghantanha, and G. Srivastava, "Federated learning-based anomaly detection for iot security attacks," *IEEE Internet of Things Journal*, 2021.

[73] E. Seo, H. M. Song, and H. K. Kim, "Gids: Gan based intrusion detection system for in-vehicle network," in *2018 16th Annual Conference on Privacy, Security and Trust (PST)*, IEEE, 2018, pp. 1–6.

[74] A. Ferdowsi and W. Saad, "Generative adversarial networks for distributed intrusion detection in the internet of things," in *2019 IEEE Global Communications Conference (GLOBECOM)*, IEEE, 2019, pp. 1–6.

[75] Z. Chkirbene, H. B. Abdallah, K. Hassine, R. Hamila, and A. Erbad, "Data augmentation for intrusion detection and classification in cloud networks," in *2021 International Wireless Communications and Mobile Computing (IWCMC)*, IEEE, 2021, pp. 831–836.

[76] S. Ganapathy, K. Kulothungan, S. Muthurajkumar, M. Vijayalakshmi, P. Yogesh, and A. Kannan, "Intelligent feature selection and classification techniques

for intrusion detection in networks: A survey," *EURASIP Journal on Wireless Communications and Networking*, vol. 2013, no. 1, p. 271, 2013.

[77] A. Tabassum, A. Erbad, and M. Guizani, "A survey on recent approaches in intrusion detection system in iots," in *2019 15th International Wireless Communications & Mobile Computing Conference (IWCMC)*, IEEE, 2019, pp. 1190–1197.

[78] K. Lee, K. Lee, H. Lee, and J. Shin, "A simple unified framework for detecting out-of-distribution samples and adversarial attacks," in *Advances in Neural Information Processing Systems*, 2018, pp. 7167–7177.

[79] Y. Chen, F. Luo, T. Li, T. Xiang, Z. Liu, and J. Li, "A training-integrity privacy-preserving federated learning scheme with trusted execution environment," *Information Sciences*, vol. 522, pp. 69–79, 2020.

[80] Y. Li, Y. Wang, Q. Liu, C. Bi, X. Jiang, and S. Sun, "Incremental semi-supervised learning on streaming data," *Pattern Recognition*, vol. 88, pp. 383–396, 2019.

[81] K. Wu, Z. Chen, and W. Li, "A novel intrusion detection model for a massive network using convolutional neural networks," *IEEE Access*, vol. 6, pp. 50 850–50 859, 2018.

[82] G. E. Hinton, S. Osindero, and Y.-W. Teh, "A fast learning algorithm for deep belief nets," *Neural computation*, vol. 18, no. 7, pp. 1527–1554, 2006.

[83] M. S. G. Karypis, V. Kumar, and M. Steinbach, "A comparison of document clustering techniques," in *TextMining Workshop at KDD2000 (May 2000)*, 2000.

[84] M. Tavallaee, E. Bagheri, W. Lu, and A. A. Ghorbani, "A detailed analysis of the kdd cup 99 data set," in *2009 IEEE symposium on computational intelligence for security and defense applications*, IEEE, 2009, pp. 1–6.

[85] O. Irsoy and E. Alpaydın, "Unsupervised feature extraction with autoencoder trees," *Neurocomputing*, vol. 258, pp. 63–73, 2017.

[86] S. Potluri and C. Diedrich, "Accelerated deep neural networks for enhanced intrusion detection system," in *2016 IEEE 21st international conference on emerging technologies and factory automation (ETFA)*, IEEE, 2016, pp. 1–8.

[87] M. Li, D. G. Andersen, J. W. Park, A. J. Smola, A. Ahmed, V. Josifovski, J. Long, E. J. Shekita, and B.-Y. Su, "Scaling distributed machine learning with the parameter server," in *11th {USENIX} Symposium on Operating Systems Design and Implementation ({OSDI} 14)*, 2014, pp. 583–598.

[88] J. Dean, G. Corrado, R. Monga, K. Chen, M. Devin, M. Mao, M. Ranzato, A. Senior, P. Tucker, K. Yang, *et al.*, "Large scale distributed deep networks," *Advances in neural information processing systems*, vol. 25, pp. 1223–1231, 2012.

[89] J. Konečn, H. B. McMahan, F. X. Yu, P. Richtárik, A. T. Suresh, and D. Bacon, "Federated learning: Strategies for improving communication efficiency," *arXiv preprint arXiv:1610.05492*, 2016.

[90] K. Bonawitz, H. Eichner, W. Grieskamp, D. Huba, A. Ingerman, V. Ivanov, C. Kiddon, J. Konečn, S. Mazzocchi, H. B. McMahan, *et al.*, "Towards federated learning at scale: System design," *arXiv preprint arXiv:1902.01046*, 2019.

[91] A. Antoniou, A. Storkey, and H. Edwards, "Data augmentation generative adversarial networks," *arXiv preprint arXiv:1711.04340*, 2017.

[92] C. Fan and P. Liu, "Federated generative adversarial learning," in *Chinese Conference on Pattern Recognition and Computer Vision (PRCV)*, Springer, 2020, pp. 3–15.

[93] M. Rasouli, T. Sun, and R. Rajagopal, "Fedgan: Federated generative adversarial networks for distributed data," *arXiv preprint arXiv:2006.07228*, 2020.

[94] C. Hardy, E. Le Merrer, and B. Sericola, "Md-gan: Multi-discriminator generative adversarial networks for distributed datasets," in *2019 IEEE international parallel and distributed processing symposium (IPDPS)*, IEEE, 2019, pp. 866–877.

[95] B. Xin, W. Yang, Y. Geng, S. Chen, S. Wang, and L. Huang, "Private fl-gan: Differential privacy synthetic data generation based on federated learning," in *ICASSP 2020-2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, IEEE, 2020, pp. 2927–2931.

[96] R. Vinayakumar, K. Soman, and P. Poornachandran, "Applying convolutional neural network for network intrusion detection," in *2017 International Conference on Advances in Computing, Communications and Informatics (ICACCI)*, IEEE, 2017, pp. 1222–1228.

[97] C. Dwork, A. Roth, *et al.*, "The algorithmic foundations of differential privacy.," *Found. Trends Theor. Comput. Sci.*, vol. 9, no. 3-4, pp. 211–407, 2014.

[98] A. Warzyński and G. Kołaczek, "Intrusion detection systems vulnerability on adversarial examples," *2018 Innovations in Intelligent Systems and Applications (INISTA)*, pp. 1–4, 2018.

[99] M. Zhang, E. Wei, and R. Berry, "Faithful edge federated learning: Scalability and privacy," *IEEE Journal on Selected Areas in Communications*, 2021.

[100] E. Bagdasaryan, A. Veit, Y. Hua, D. Estrin, and V. Shmatikov, "How to backdoor federated learning," in *International Conference on Artificial Intelligence and Statistics*, PMLR, 2020, pp. 2938–2948.

[101] S. Chen, M. Xue, L. Fan, S. Hao, L. Xu, H. Zhu, and B. Li, "Automated poisoning attacks and defenses in malware detection systems: An adversarial machine learning approach," *computers & security*, vol. 73, pp. 326–344, 2018.

[102] T. Yu, E. Bagdasaryan, and V. Shmatikov, "Salvaging federated learning by local adaptation," *arXiv preprint arXiv:2002.04758*, 2020.

[103] M. Nasr, R. Shokri, and A. Houmansadr, "Comprehensive privacy analysis of deep learning: Passive and active white-box inference attacks against centralized and federated learning," in *2019 IEEE symposium on security and privacy (SP)*, IEEE, 2019, pp. 739–753.

[104] Y. Zhao, J. Chen, J. Zhang, D. Wu, J. Teng, and S. Yu, "Pdgan: A novel poisoning defense method in federated learning using generative adversarial network," in *International Conference on Algorithms and Architectures for Parallel Processing*, Springer, 2019, pp. 595–609.

[105] H. Moustafa, E. M. Schooler, G. Shen, and S. Kamath, "Remote monitoring and medical devices control in ehealth," in *2016 IEEE 12th International Conference*

on Wireless and Mobile Computing, Networking and Communications (WiMob), IEEE, 2016, pp. 1–8.

[106] A. Tabassum and W. Lebda, "Security framework for iot devices against cyber-attacks," *arXiv preprint arXiv:1912.01712*, 2019.

[107] G. De La Torre, P. Rad, and K.-K. R. Choo, "Driverless vehicle security: Challenges and future research opportunities," *Future Generation Computer Systems*, 2018.

[108] S. Meggitt, "Medjack attacks: The scariest part of the hospital," 2018.

[109] N. Davies, N. Taft, M. Satyanarayanan, S. Clinch, and B. Amos, "Privacy mediators: Helping iot cross the chasm," in *Proceedings of the 17th International Workshop on Mobile Computing Systems and Applications*, ACM, 2016, pp. 39–44.

[110] H. Fotouhi, A. Causevic, K. Lundqvist, and M. Björkman, "Communication and security in health monitoring systems – a review," in *2016 IEEE 40th Annual Computer Software and Applications Conference (COMPSAC)*, vol. 1, 2016, pp. 545–554.